Cognitive Complexity

Yegor Bugayenko

Lecture #3 out of 24
80 minutes

The slidedeck was presented by the author in this YouTube Video

1. Cyclomatic complexity is highly inaccurate if being compared with code readability evaluated by a human.

Bill Curtis

"Computational complexity assesses the difficulty of verifying an algorithm's correctness, while psychological complexity assesses human performance on programming tasks."

— Bill Curtis, Sylvia B. Sheppard, Phil Milliman, M. A. Borst, and Tom Love. Measuring the Psychological Complexity of Software Maintenance Tasks With the Halstead and McCabe Metrics. *IEEE Transactions on Software Engineering*, 5 (2):96–104, 1979. doi:10.1109/TSE.1979.234165

2. There must be other way to measure the complexity of code, taking into account how humans comprehend it.

Lionel C. Briand

"By cognitive complexity we mean the mental burden of the persons who have to deal with the class. We assume that it is the, sometimes necessary, high cognitive complexity of a class which causes it to display undesirable external qualities, such as decreased maintainability and testability, or increased fault-proneness."

— Lionel C. Briand and Jürgen Wüst. Integrating Scenario-Based and Measurement-Based Software Product Assessment. *Journal of Systems and Software*, 59(1):3–22, 2001. doi:10.1016/S0164-1212(01)00045-0

Jingqiu Shao

"Cognitive complexity, the new measure for software complexity presented in this paper, is a measure of the cognitive and psychological complexity of software as a human intelligence artifact."

— Jingqiu Shao and Yingxu Wang. A New Measure of Software Complexity Based on Cognitive Weights. *Canadian Journal of Electrical and Computer Engineering*, 28(2):69–74, 2003. doi:10.1109/CJECE.2003.1532511

Table 1
Definition of BCSs and their equivalent cognitive weights (W_i)

Category	BCS	Structure	W_i	RTPA notation
Sequence	Sequence (SEQ)		1	P→Q Note: Consider only one sequential structure in a component
Branch	If-then-[else] (ITE)		2	(?exp **BL** = **T**)→P | (?~)→Q
	Case (CASE)		3	? exp **RT** = 0→P_0 | 1→P_1 | … | n–1→P_{n-1} | else→Ø
Iteration	For-do (R_i)		3	$R_{i=1}^{n}(P(i))$
	Repeat-until (R_1)		3	$R_{\geq 1}^{\text{exp BL}\neq\text{T}}(P)$

BCS stands for "basic control structure."

"The cognitive weight of software is the degree of difficulty or relative time and effort required for comprehending a given piece of software modelled by a number of BCSs. The total cognitive weight of a software component, W_c, is defined as the sum of the cognitive weights of its q linear blocks composed of individual BCSs."

Source: Jingqiu Shao and Yingxu Wang. A New Measure of Software Complexity Based on Cognitive Weights. *Canadian Journal of Electrical and Computer Engineering*, 28(2):69–74, 2003. doi:10.1109/CJECE.2003.1532511

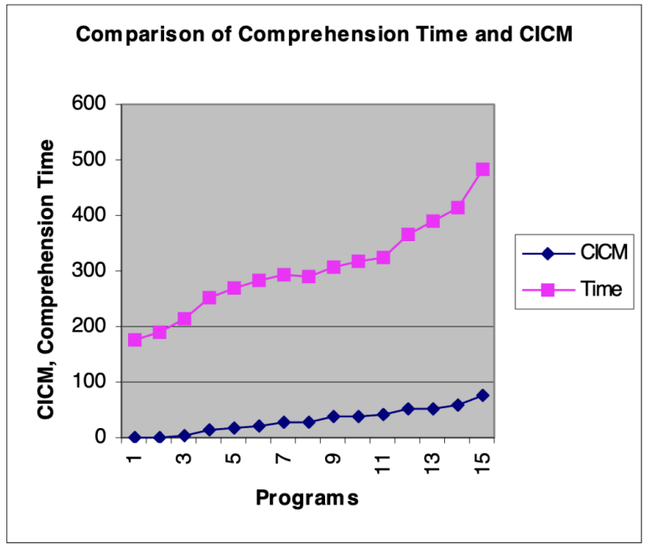
“Cognitive complexity of software is a product of its architectural and operational complexities on the basis of deductive semantics and the abstract system theory.”

— Yingxu Wang. Cognitive Complexity of Software and Its Measurement. In *Proceedings of the 5th International Conference on Cognitive Informatics*, volume 1, pages 226–235. IEEE, 2006. doi:10.1109/COGINF.2006.365701

Dharmender Singh Kushwaha

"We developed an improved cognitive information complexity measure (CICM) that is based on the amount of information contained in the software and encompasses all the major parameters that have a bearing on the difficulty of comprehension or cognitive complexity of software."

— Dharmender Singh Kushwaha and Arun Kumar Misra. Improved Cognitive Information Complexity Measure: A Metric That Establishes Program Comprehension Effort. *ACM SIGSOFT Software Engineering Notes*, 31(5):1–7, 2006. doi:10.1145/1163514.1163533



"In a group of 25, each student was given 15 programs. They were asked to read the program and come out with what problem the program addressed. Time required to understand was recorded. The diagram shows that as cognitive information complexity increases, time taken to understand the program also increased."

Source: Dharmender Singh Kushwaha and Arun Kumar Misra. Improved Cognitive Information Complexity Measure: A Metric That Establishes Program Comprehension Effort. *ACM SIGSOFT Software Engineering Notes*, 31(5):1–7, 2006. doi:10.1145/1163514.1163533

“The cognitive weight of a BCS is measured as the number of ways that BCS can generate some factors that make it difficult to comprehend relative to the sequential BCS, of which the cognitive weight is ‘1’. The value of total cognitive weights of the software is measured as the number of relative ways that software can generate the combination of factors that make the function and semantics difficult to comprehend.”

— Auprasert Benjapol and Yachai Limpiyakorn. Underlying Cognitive Complexity Measure Computation With Combinatorial Rules, 2008

3. One metric was adopted by the community and is actively used.

sonarqube

Ann Campbell

"A guiding principle in the formulation of Cognitive Complexity has been that it should incent good coding practices. That is, it should either ignore or discount features that make code more readable."

— G. Ann Campbell. Cognitive Complexity: An Overview and Evaluation. In *Proceedings of the International Conference on Technical Debt*, pages 57–58, 2018. doi:10.1145/3194164.3194186

```
int sumOfPrimes(int max) {                  // +1
  int total = 0;
  OUT: for (int i = 1; i <= max; ++i) {     // +1
    for (int j = 2; j < i; ++j) {           // +1
      if (i % j == 0) {                     // +1
        continue OUT;
      }
    }
    total += i;
  }
  return total;
}                  // Cyclomatic Complexity 4
```

```
String getWords(int number) { // +1
    switch (number) {
      case 1:                 // +1
        return "one";
      case 2:                 // +1
        return "a couple";
      case 3:                 // +1
        return "a few";
      default:
        return "lots";
    }
  }     // Cyclomatic Complexity 4
```

"The mathematical model underlying Cyclomatic Complexity gives these two methods equal weight, yet it is intuitively obvious that the control flow of `sumOfPrimes` is more difficult to understand than that of `getWords`."

Source: G. Ann Campbell. {Cognitive Complexity} — A New Way of Measuring Understandability, 2017

```
int sumOfPrimes(int max) {
  int total = 0;
  OUT: for (int i = 1; i <= max; ++i) { // +1
    for (int j = 2; j < i; ++j) {       // +2
      if (i % j == 0) {                 // +3
        continue OUT;                   // +1
      }
    }
    total += i;
  }
  return total;
}                  // Cognitive Complexity 7
```

```
String getWords(int number) {
    switch (number) {          // +1
      case 1:
        return "one";
      case 2:
        return "a couple";
      case 3:
        return "a few";
      default:
        return "lots";
    }
  }      // Cognitive Complexity 1
```

"The Cognitive Complexity algorithm gives these two methods markedly different scores, ones that are far more reflective of their relative understandability."

Source: G. Ann Campbell. {Cognitive Complexity} — A New Way of Measuring Understandability, 2017

Rubén Saborido

"Different cognitive complexity measures have been proposed to quantify the understandability of a piece of code and, therefore, its maintainability. However, the cognitive complexity metric provided by SonarSource is quickly spreading in the software industry due to the popularity of their well-known static code tools for evaluating software quality."

— Rubén Saborido, Javier Ferrer, Francisco Chicano, and Enrique Alba. Automatizing Software Cognitive Complexity Reduction. *IEEE Access*, 10(1): 11642–11656, 2022. doi:0.1109/ACCESS.2022.3144743

Cognitive Complexity (CoCo) is supported by a few static analyzers:

- for Java by PMD since 6.22.0
- for Ruby by Rubocop since 0.25 (called “perceived complexity”)
- for C++ by CodeClimate
- for JavaScript by ESLint via SonarSource plugin
- for Python by Flake8 via this plugin
- for Rust by Clippy since 1.35.0
- for PHP by PHPCS via this plugin

4. Maybe it's impossible to accurately measure the "perceived complexity" of software?

"We showed that the search for general software complexity measures is doomed to failure."

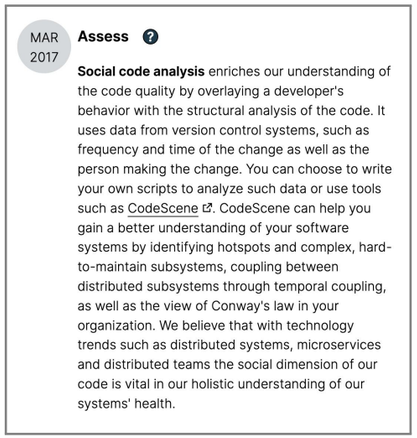
— Norman Fenton. Software Measurement: A Necessary Scientific Basis. *IEEE Transactions on Software Engineering*, 20(3):199–206, 1994. doi:10.1109/32.268921

Luigi Lavazza

"Being able to find statistical correlations between code understandability and source code measures would be greatly beneficial for the software development process, which involves a great deal of activities involving program comprehension."

— Luigi Lavazza, Abedallah Zaid Abualkishik, Geng Liu, and Sandro Morasca. An Empirical Evaluation of the "Cognitive Complexity" Measure as a Predictor of Code Understandability. *Journal of Systems and Software*, 197(1), 2023. doi:10.1016/j.jss.2022.111561

5. Social code analysis: we analyze the author(s) of the code, not only the code itself.


MAR 2017

Assess

Social code analysis enriches our understanding of the code quality by overlaying a developer's behavior with the structural analysis of the code. It uses data from version control systems, such as frequency and time of the change as well as the person making the change. You can choose to write your own scripts to analyze such data or use tools such as CodeScene. CodeScene can help you gain a better understanding of your software systems by identifying hotspots and complex, hard-to-maintain subsystems, coupling between distributed subsystems through temporal coupling, as well as the view of Conway's law in your organization. We believe that with technology trends such as distributed systems, microservices and distributed teams the social dimension of our code is vital in our holistic understanding of our systems' health.

"Social code analysis enriches our understanding of the code quality by overlaying a developer's behavior with the structural analysis of the code."

— ThoughtWorks. Social Code Analysis. https://jttu.net/social2017, 3 2017. [Online; accessed 09-10-2024]

“BrainMaster Technologies (USA) sells this “Freedom 20R” EEG head set for $31,025. A similar device you can get from Neiry (Russia), for 3% of this price.”

"We propose the use of biofeedback code highlighting techniques to provide online programmer's warnings for potentially problematic code lines that may need a second look at (to remove possible bugs)."

— Ricardo Couceiro, Raul Barbosa, João Durães, Gonçalo Duarte, João Castelhano, Catarina Duarte, Cesar Teixeira, Nuno Laranjeiro, Júlio Medeiros, Paulo Carvalho, et al. Spotting Problematic Code Lines Using Nonintrusive Programmers' Biofeedback. In *Proceedings of the 30th International Symposium on Software Reliability Engineering (ISSRE)*, pages 93–103. IEEE, 2019. doi:10.1109/ISSRE.2019.00019

The following Java code has an obvious but hard to spot functional defect:

```
1  public class AverageCalculator {                          // LOW
2    public static void main(String[] args) {                // LOW
3      int[] numbers = {10, 20, 30, 40, 50};                 // LOW
4      System.out.println("Average: " + avg(numbers));       // MEDIUM
5    }                                                       // LOW
6    static double avg(int[] nums) {                         // LOW
7      double sum = 0;                                       // MEDIUM
8      for (int i = 0; i <= nums.length; i++)                // HIGH
9        sum += nums[i];                                     // HIGH
10     return sum / nums.length;                             // MEDIUM
11   }                                                       // LOW
12 }                                                         // LOW
```

The “cognitive state code annotations” are on the right side of the code, as Java comments, indicating the level of brain activity at the moment of writing the particular line of code.

"The EEG results reveal evidence of mental effort saturation as code complexity increases. Conversely, the classic software complexity metrics do not accurately represent the mental effort involved in code comprehension."

— Júlio Medeiros, Ricardo Couceiro, Gonçalo Duarte, João Durães, João Castelhano, Catarina Duarte, Miguel Castelo-Branco, Henrique Madeira, Paulo de Carvalho, and César Teixeira. Can EEG Be Adopted as a Neuroscience Reference for Assessing Software Programmers' Cognitive Load? *Sensors*, 21(7): 2338, 2021. doi:10.3390/s21072338

Bibliography

Auprasert Benjapol and Yachai Limpiyakorn. Underlying Cognitive Complexity Measure Computation With Combinatorial Rules, 2008.

Lionel C. Briand and Jürgen Wüst. Integrating Scenario-Based and Measurement-Based Software Product Assessment. *Journal of Systems and Software*, 59(1):3–22, 2001. doi:10.1016/S0164-1212(01)00045-0.

G. Ann Campbell. {Cognitive Complexity} — A New Way of Measuring Understandability, 2017.

G. Ann Campbell. Cognitive Complexity: An Overview and Evaluation. In *Proceedings of the International Conference on Technical Debt*, pages 57–58, 2018. doi:10.1145/3194164.3194186.

Ricardo Couceiro, Raul Barbosa, João Duráes, Gonçalo Duarte, João Castelhano, Catarina Duarte, Cesar Teixeira, Nuno Laranjeiro, Júlio Medeiros, Paulo Carvalho, et al. Spotting Problematic Code Lines Using Nonintrusive Programmers' Biofeedback. In *Proceedings of the 30th International Symposium on Software Reliability Engineering (ISSRE)*, pages 93–103. IEEE, 2019. doi:10.1109/ISSRE.2019.00019.

Bill Curtis, Sylvia B. Sheppard, Phil Milliman, M. A. Borst, and Tom Love. Measuring the Psychological Complexity of Software Maintenance Tasks With the Halstead and McCabe Metrics. *IEEE Transactions on Software Engineering*, 5(2):96–104, 1979. doi:10.1109/TSE.1979.234165.

Norman Fenton. Software Measurement: A Necessary Scientific Basis. *IEEE Transactions on Software Engineering*, 20(3):199–206, 1994. doi:10.1109/32.268921.

Dharmender Singh Kushwaha and Arun Kumar Misra. Improved Cognitive Information Complexity Measure: A Metric That Establishes Program Comprehension Effort. *ACM SIGSOFT Software Engineering Notes*, 31(5): 1–7, 2006. doi:10.1145/1163514.1163533.

Luigi Lavazza, Abedallah Zaid Abualkishik, Geng Liu, and Sandro Morasca. An Empirical Evaluation of the "Cognitive Complexity" Measure as a Predictor of Code Understandability. *Journal of Systems and Software*, 197 (1), 2023. doi:10.1016/j.jss.2022.111561.

Júlio Medeiros, Ricardo Couceiro, Gonçalo Duarte, João Durães, João Castelhano, Catarina Duarte, Miguel Castelo-Branco, Henrique Madeira, Paulo de Carvalho, and César Teixeira. Can EEG Be Adopted as a Neuroscience Reference for Assessing Software Programmers' Cognitive Load? *Sensors*, 21(7):2338, 2021. doi:10.3390/s21072338.

Rubén Saborido, Javier Ferrer, Francisco Chicano, and Enrique Alba. Automatizing Software Cognitive Complexity Reduction. *IEEE Access*, 10(1):11642–11656, 2022. doi:0.1109/ACCESS.2022.3144743.

Jingqiu Shao and Yingxu Wang. A New Measure of Software Complexity Based on Cognitive Weights. *Canadian Journal of Electrical and Computer Engineering*, 28(2):69–74, 2003. doi:10.1109/CJECE.2003.1532511.

ThoughtWorks. Social Code Analysis. `https://jttu.net/social2017`, 3 2017. [Online; accessed 09-10-2024].

Yingxu Wang. Cognitive Complexity of Software and Its Measurement. In *Proceedings of the 5th International Conference on Cognitive Informatics*, volume 1, pages 226–235. IEEE, 2006. doi:10.1109/COGINF.2006.365701.