ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 7, Issue 12, December 2019

Data Analytics Tools for Software Complexity Metrics: A Comparative Study

Kehinde SOTONWA[1], Maria ADIGUN[2]

P.G. Student, Department of Computer Engineering and Information Technology, Bells University of technology, Ota, Ogun State, Nigeria[1]

Technologist II, Department of Computer Science and Information Technology, Bells University of Technology, Ota, Ogun State, Nigeria[2]

ABSTRACT: Data science is an essential part of technology industry revived due to increase in computing power, presence of huge amounts of data and better understanding techniques in the area of data analytics, artificial intelligence, machine learning and deep learning for solving many challenging problems. In the search for a good programming language on which many data science applications can be developed, the need to develop quality and cost effective software cannot be overemphasized. Hence, there arises the need to apply code based metrics to three different data analytical tools (Python, R and Scala) to evaluate the complexities of different implementation of quick sort algorithm and measure the degree of relationship among them. It was discovered that Scala is realized to be the most composite tool for all the metrics while Python and R are averagely at the same level.

KEYWORDS*:* Code based metrics;quicksort algorithm;Python; R and Scala

I. INTRODUCTION

The demand for data scientists in every industry is growing substantially; for the development of every business, there is a need to assess the data gathered while data scientists require both the right tools and perfect skill set to enable better results with information. Data analytics is the science of analyzing raw data in order to make conclusions about that information. Many of the techniques and processes of data analytics have been automated into mechanical processes and algorithms that work over raw data for human consumption.Data analytics techniques can reveal trends and metrics that would otherwise be lost in the mass of information. This information can then be used to optimize processes to increase the overall efficiency of a business or system. Data analysis has multiple facets and approaches, encompassing diverse techniques under a variety of names, and is used in different business, science, and social science domains **[1, 2].** Among the popular tools used for data analytics include Tensorflow, Java, SQL, MATLAB, Python, R, Scala, Julia, SAS**[3, 4].**It is always hard to control software quality if the code is complex. Complex codes always create problems for software communities as it is hard to review, test, maintain as well as manage such codes. The revival of data science due to the presence of large amount of data has resulted in the need for a good programming language on which many data science applications can be developed**[5].**

In software engineering, code based metrics are the only tools to control the quality of software **[6].**Code based metrics determine the degree of maintainability of software products, which is one of the important factors that affect the quality of any kind of software. It provide useful feedback to the designers to impact the decisions that are made during design, coding, architecture, or specification phases which without such feedback, many decisions are made in an havoc manner**[7].**Software life cycle is the process of developing and changing software systems. A software life cycle consists of all the activities and products that are needed to develop a software system. Due to the fact that software systems are complex, life cycle models tend to enable developers to cope with software complexity. Life cycle models

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 7, Issue 12, December 2019

expose the software development activities and their dependencies in order to make them more visible and manageable **[8. 9, 10, 11].**

The kinds of complexities that may be encountered in software engineering includes architectural complexity, cognitive complexity, component and time complexity, control flow complexity, computational complexity, data scope complexity, functional complexity, inheritance complexity, program complexity, problem complexity, system complexity, syntactic complexity and programming/coding/software complexity.

Sorting is a process of rearranging a list of elements to the correct order since handling the elements in a certain order is more efficient than handling randomized elements **[12].** The rapid growth of data and information has led to an increase in research and formulation of divers sort algorithms. Developing sort algorithms to improve performance and decrease complexity has attracted a great deal of research**[13, 14].** Among the algorithm used for large data is sorting algorithm and quicksort is the fastest among the type of sorting algorithm. Quick sort is a divide and conquer algorithm which relies on a partition operation: to partition an array an element called a pivot is selected **[15].** All elements smaller than the pivot is moved before it and all greater elements are moved after it. This can be done efficiently in linear time and in-place. The lesser and greater sub lists are then recursively sorted**[16].**

II. RELATED WORK

In **[17]**authors made comparison between the grouping comparison sort (GCS) and conventional algorithm on selection sort, quick sort, insertion sort, merge sort and bubble sort with respect execution time and it was discovered that for large input quick sort is the fastestwhile the future pose on optimizing software in searching method and retrieving data. Authors**[18]** evaluated the performance of media, heap and quick sort techniques using CPU time and memory space as performance index, implemented in C language and it was discovered that the slowest technique is media sort while quick sort is faster and required less memory and future work is posed on adopting the most efficient sorting technique in developing job scheduler for grid computing community.

Authors **[19]**compared software complexity of Line of Code, cyclomatic complexity metric and Halstead complexity metrics of linear and binary search algorithms using VB, C#, C++ and Java to measure the sample programs using length in lines of the program, LOC with comments, LOC without comments, McCabe method and the program difficulty using Halstead method. It was discovered that the four object-oriented programming languages is good to code linear and binary search algorithms. However, procedural language can also be applied on software metrics. Authors **[20]** performed a quantitative analysis on experiments utilizing three different tools (Python R and SAS) used for data science which include replication of analysis along with comparisons of code length, output and result to supplement the quantitative findings. The comparative analysis did not identify a single tool for all circumstance while the experiment showed situation where each tool performed better than the others with strength and weaknesses for various activities therefore, it was discovered that there is provision of data support guidance on the correct tool to use for common situations in the field of data science.

In**[21]** a multi-paradigm complexity metric (MCM) for measuring software complexity of C++ and Python which combine the features of procedural and object oriented paradigms was proposed. The developed metric was applied on software complexity metrics (eLOC, cyclomatic complexity metric and Halstead measures). It was discovered that the developed metric have significant comparison with the existing complexity measures and can be used to rank numerous program and difficulty of various modules. However of all the types of LOC only eLOC was used; Future work may be geared towards evaluating data analytical language on software measures.Authors **[22]** matched rigorous object oriented application (VB, C#, C++, Java and Python) languages of linear and binary search algorithms using software complexity metrics (LOC, McCabe method and Halstead method) which allowed for consistency in the object oriented languages. Statistical evaluation was performed on the metrics using Person correlation coefficient which showed a high degree of correlation existence among (VB, C#, C++, Java and Python) and analysis of variance (ANOVA) showed that for VB, C#, C++, Java and Python is good to code linear andbinary search algorithms. However, other object oriented programming languages can be used to justify this result

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 7, Issue 12, December 2019

III. PROPOSED ALGORITHM

A. *Lines of Code (LOC):*

The line of codes (LOC) is generally considered as the count of the lines in the source code of the software. Usually, (LOC) only considers the executable sentence. LOC is independent of what program language used. The LOC evaluates the complexity of the software via the physical length **[23, 24].**The original purpose of its development was to estimate man-hours for a project **[25].**

- Counts every line of the program including comments, standalone brace, blank lines and parenthesis.

B. *Halstead Complexity Metric:*

Maurice Howard Halstead (1977) introduced the concept of software science. He began to use scientific methods to analyze the characteristics and structure of the software. The idea resulted in the introduction of the Halstead complexity metric (HCM)**[26].** The HCM is calculated on the count of the operators and operands. The operators are symbols used in expressions to specify the manipulations to be performed**[27].** The operands are the basic logic unit to be operated. The HCM measures the logic volume and compute the following parameters:

μ_1= the number of unique operators
μ_2= the number of unique operands
N_1 = the total occurrences of operators
N_2= the total occurrences of operands

Step 1: Calculating Length N of the Program
Using Halstead method eq.(1)[8].

$$N = N_1 + N_2 \quad \text{eq. (1)}$$

Step 2: The vocabulary μ of P:

$$\mu = \mu_1 + \mu_2 \quad \text{eq. (2)}$$

Step 3: Program Difficulty: using Halstead Method:

$$Dof\ P = (\mu_1 \div 2) * (N_2 \div \mu_2) \quad \text{eq. (3)}$$

Step 4: Volume using Halstead Method:

$$V = N *_{log_2} (\mu)_1 \quad \text{eq. (4)}$$

Step 5: Effort: E to Generate Program is calculated using Halstead Method:

$$E = D * V \quad \text{eq. (5)}$$

where D is the difficulty and V is the volume

Step 6: Number of Bugs:

$$B = E^{(2/3)}/3000 \quad \text{eq. (6)}$$

Step 7: Error: using Halstead method

$$B = V/_{X*} \quad \text{eq. (7)}$$

where B, is the number of delivered bugs, V is the volume of the program and Halstead sets X* for a fixed value of 3000.

$$B = E^{(2/3)}/3000 \quad \text{eq. (8)}$$

Step 8: Time: using Halstead method

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 7, Issue 12, December 2019

$$T = {}^{E}/_{18} \quad \text{eq. (9)}$$

where E is the Effort to generate program

C. *McCabe Cyclomatic Complexity Metric:*

Based upon the topological structure of the software, Thomas J. McCabe introduced a software complexity metric named McCabe Cyclomatic Complexity Metric. As described by McCabe, the primary purpose of the measure is to identify software modules that will be difficult to test or maintain **[28].** The nodes of the graph correspond to the code lines of the software, and a directed edge connects two nodes if the second node might be executed immediately after the first one. If the conditional evaluation expression is composite, the expression should be broken down.

Step 1: Calculating McCabe method using cycomatic complexity method**[29].**

$$MC = V(G) = e - n + 2p \quad \text{eq. (10)}$$

where e is the edges, n is the nodes and p is the connected component

IV. PSEUDO CODE

The sort algorithm for quick sort is based on the effort required in understanding the software the information contained from the codes of Python, R and Scala considered respectively:

Step 1: choose the highest index values as pivot
Step 2: take two variables to point left and right of the list excluding pivot
Step 3: left points to the low index
Step 4: right points to the high.
Step 5: while value at left is less than pivot move right.
Step 6: while vales at right is greater than pivot move left.
Step 7: if both step5 and 6 do not match swap left and right
Step 8: if left ≥ right, the point where they met is new pivot
Step 9: end.

V. SIMULATION RESULTS

The Figures 1, 3 and 6 showed the quick sort algorithms in Python, R and Scala while Figures 2, 4 and showed the flow graph representation for the quick sort algorithms for Python, R and Scala.

```
def quicksort1(q):
    """
    The basic version.
    http://zh.wikipedia.org/wiki/%E5%BF%AB%E9%80%9F%E6%8E%92%E5%BA%8F#.E6.BC.94.E7.AE.97.E6.B3.95
    """
    less = []
    pivotList = []
    greater = []
    length = len(q)
    if length <= 1:
        return q
    else:
        pivotIndex = length // 2 - 1;
        pivot = q[pivotIndex]
        i = -1
        for x in q:
            i += 1
            if i == pivotIndex:
                continue
            if x < pivot:
                less.append(x)
            else:
                greater.append(x)
        pivotList.append(pivot)
        return quicksort1(less) + pivotList + quicksort1(greater)
```

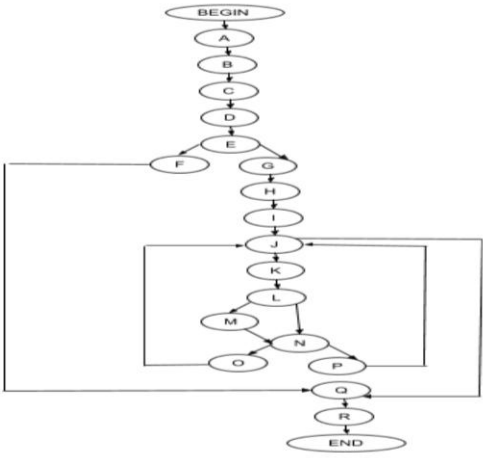

Figure 1.Quicksort Algorithm written in Python Figure 2. Flow Graph Representation of Quicksort Algorithm in Python

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 7, Issue 12, December 2019

```
quickSort <- function(arr) {
  # Pick a number at random.
  mid <- sample(arr, 1)
  # Place-holders for left and right values.
  left <- c()
  right <- c()
  # Move all the smaller values to the left, bigger values to the right.
  lapply(arr[arr != mid], function(d) {
    if (d < mid) {
      left <<- c(left, d)
    }
    else {
      right <<- c(right, d)
    }
  })
  if (length(left) > 1) {
    left <- quickSort(left)
  }
  if (length(right) > 1) {
    right <- quickSort(right)
  }
  # Finally, return the sorted values.
  c(left, mid, right)
}
```

Figure 3. Quicksort Algorithm written in R

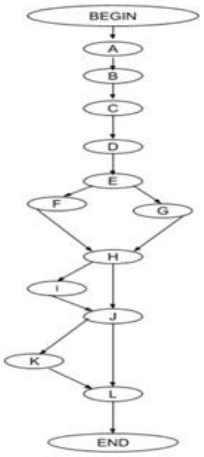

Figure 4. Flow Graph Representation of Quicksort Algorithm in R

```
    while (i < j) {
    while (array(i) <= array(pivot) && i < last) {
      i += 1
    }
    while (array(j) > array(pivot)) {
      j -= 1
    }
    if (i < j) {
      temp = array(i)
      array(i) = array(j)
      array(j) = temp
    }
  }
  temp = array(pivot)
  array(pivot) = array(j)
  array(j) = temp
  quickSortImpl(array, first, j - 1)
  quickSortImpl(array, j + 1, last)
}
array
}
quickSortImpl(array,0, array.length-1)
}
}
```

Figure 5. Quicksort Algorithm written in Scala

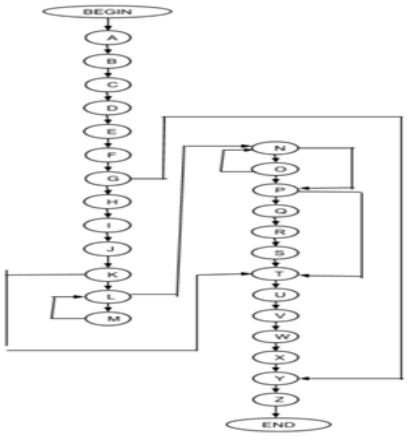

Figure 6. Flow Graph Representation of Quicksort Algorithm in Scala

Table 1 displayed the complexities measures for McCabe, error estimate and program difficulty for Python, R and Scala, so also Table 2 demonstrated all the complexities vales for line of codes, programming time and volume for Python, R and Scala analytical languages.

Table 1: Complexities Analysis of Quicksort Algorithm

Complexity Type	Python	R	Scala
McCabe	5	4	6
Error Estimate	0.258	0.199	0.495
Program Difficulty	8.56	13.84	15.96

Table 2: Complexities Analysis of Quicksort Algorithm

Complexity Type	Python	R	Scala
Line of Codes	25	24	41
Programming Time	1072	775	3955.95
Volume	411	595	846

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 7, Issue 12, December 2019

The Figures 7 and 8 real that for McCabe, R has lesser complexity in terms of control flow and Scala has the highest complexity value. Scala is also depicted to generate more errors, while in terms of program difficulty, it is the most difficult and R and Python are closely related in line of codes. Of the three languages, Scala has the highest volume which requires the highest programming time.

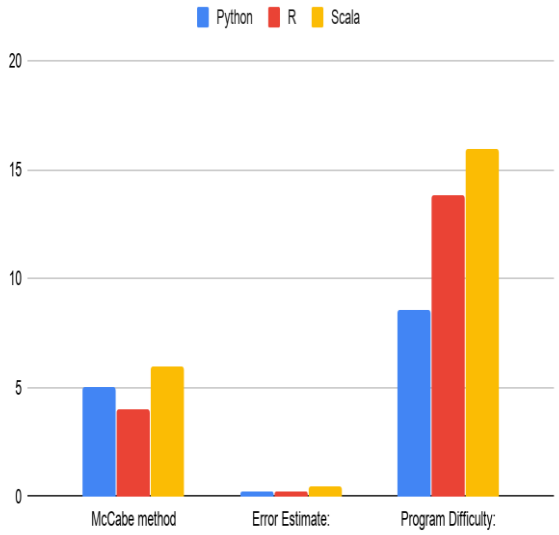

Fig.7.Relative Graph of Quicksort Algorithm: McCabe Method, Estimate and Program Difficulty

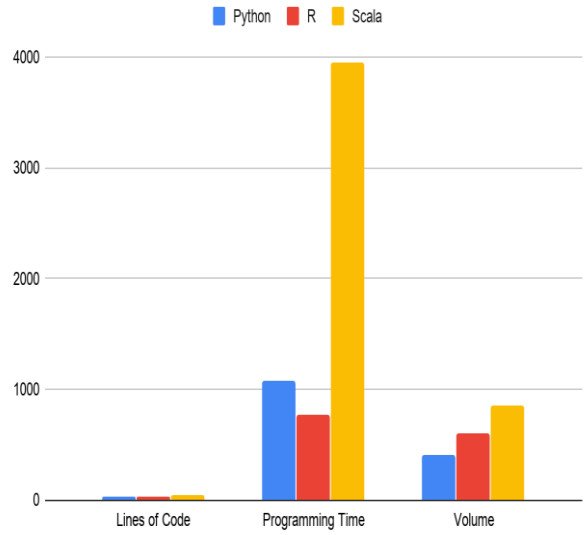

Fig.8. Relative Graph of Quicksort Algorithm: Line of Codes, Programming Time and Volume

VI. CONCLUSION AND FUTURE WORK

The code-based complexity metrics are used to quantify a variety of software properties. Complexity measures can be used to predict critical information about testability, reliability, and maintainability of the software systems from automatic analysis of the source code. It plays a vital role to reduce the effort required in maintaining software, the effectiveness of testing and software quality. The more complex the software solution, the more errors it generates. It was found that for the three complexity metrics applied, Scala has the highest complexity value as compared to Python, R, and Scala. The metrics, however, do not give the same values. Data analytic languages provide a way to break large and difficult to manage big data project into smaller modules that can be managed easily.This is because each method covers just a part and considers some parameters while leaving some others.Further research is recommended in formulating cognitive complexity measureon data analytic tools to cover the parts and parameters not covered by the existing metrics.

REFERENCES

1. Xia, B. S., and Gong, P. 'Review of Business Intelligence through Data Analysis'. Benchmarking, Vol 21 Issue 2, pp. 300-311. doi:10.1108/BIJ-08-2012-0050, 2015.
2. Andersson, M.; Vestergren, P. 'Object-Oriented Design Quality Metrics', Uppsala Master's Theses in Computer Science 276, 2004-06-07, ISSN 1100-1836, 2004.
3. Rao V. S. ' Best Programming Languages for Data Science: Artificial Intelligence Programming, 2018.
4. Akiwatkar R. 'TheMosr Popular Languagesfor Data Science: Big Data Zone- Opinion, Apr. 2017.
5. Woodie A. 'Which Programming is Best for Big Data: Datanami, A New Era Need New Storage, Hawlett Packard Enterprise Company, CRAY, 2018.
6. Andersson, M.; Vestergren, P. 'Object-Oriented Design Quality Metrics', Uppsala Master's Theses in Computer Science 276, 2004-06-07, ISSN 1100-1836, 2004.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 7, Issue 12, December 2019

7. Misra S. and Ferid C. 'Estimating Complexity of Programs in Python Language'. https://www.researchgate.net/publication/289638932_Estimating_complexity_of_programs_in_python_language
8. Bruegge Bernd, Stephan Krusche and Lukas Alperowit'Tutorial: How to Run a Multi Customer Software Engineering Capstone Course'.In 11th International Conference on Model Driven Engineering Languages and Systems, Valencia, Spain, 2014.
9. Bruegge Bernd, Stephan Krusche and Martin Wagners'Teaching Tornado : from Communication Model to Release'. In Proceedings of the 8th Edition of the Educations Symposium ACM Imsbruck, Austri, pp. 5-12, 2012.
10. Bruegge Bernd and Allen H. Dutoit 'Object Oriented Software Engineering using UML Patterns and Java, 3rd edition ed Published by Boston Prentice Hall XXX111pp. 778, ISBN: 0136061257, 9780136061250, Dewey No. 004, 2010.
11. Charles P. Pfleeger, Shari Lawrence Pfleeger and Willis H. Ware (2006): Security in Computing 4th Edition Published by Prentice Hall Oct. 27th 2006 ISBN-10: 0132390779.
12. Cormen, T. H., Leiserson, C. E., Rivest, R. L. and Stein C. ' Introduction to Algorithms: Quicksort (3rd ed.), Cambridge, MA: The MIT Press and McGraw-Hill, pp. 171–172, ISBN 978-0262033848, 2009.
13. Huang Z. S. Kannan and Khanna S. 'Algorithms for the Generalized Sorting Problem', *In Foundations of Computer Science (FOCS)*, IEEE 52nd Annual Symposium, 2011.
14. Demuth H. B. 'Electroni Data Sorting', IEEE Transactions on Computers, Vol 100, Issue 4, pp. 296-310, 1985.
15. Grover, Deepti and Beniwal S. 'Performance Analysis of Merge Sort and Quick Sort: MQSORT', 2016
16. Yadav, Neelam and Kumar S. 'Sorting Algorithms, *International Research Journal of Engineering and Technology (IRJET)*, Vol 03, Issue 12, pp. 326-330, 2016
17. Al-Kharanbsheh K. S.., AlTurani I. M., AlTurani A. I. and Zanoon N. I. 'Review on Sorting Algorithms: A Comparative Study', *International Journal of Computer Science and Security (IJCSS)*, Vol 7, Issue 3, pp. 120-126, 2013.
18. Aremu D. R., Adesina O. O., Makinde O. E., Ajibola O. and Ago-Ajala O. O. 'A Comparative Study of Sorting Algorithms', *African Journal of Computing & ICT Reference Format*, Vol 6, Issue 5, pp. 199-206, Dec., 2013.
19. Sotonwa K. A., Olabiyisi S.O, and Omidiora E. O. 'Comparative Analysis of Software Complexity of Searching Algorithms Using Code Based Metrics'. *International Journal of Scientific & Engineering Research*, Vol 4, Issue 6, pp. 2983-2993, 2013
20. Brittain J., Cendon M., Nizzi J. and Pleis J. 'Data Scientist Analysis toolbox: Comparison of Python, R and SAS Performance', SMU Data Science Review, Vol 1 No. 2, pp. 1-19, 2018.
21. Balogun M. O. and Sotonwa K. A. 'A Comparative Analysis of Complexity of C++ and Python Programming Languages Using MultiParadigm Complexity Metric (MCM)'. *International Journal of Science and Research (IJSR)* ISSN: 2319-7064, Volume 8 Issue 1, January 2019.
22. SotonwaK.,, Balogun M., Isola E., Olabiyisi S., Omidiora E. and Oyeleye C. 'Object Oriented Programming Languages for Search Algorithms in Software Complexity Metrics'. *International Research Journal of Computer Science (IRJCS)* Issue 04, Vol 6,pp.90-101, 2019.
23. NASA 'Repositor Overview, http://mdp.ivv.nasa.gov/repository.html, 2008.
24. Milutin, A. 'Software Code Metrics', (Online: accessed on 2010-06-21 from Introduction to Algorithms), 2009.
25. Sotonwa K. A., Olabiyisi S. O., Omidiora E. O and Oyeleye C. A. 'SLOC Metric in RNG Schema Documents',*International Journal of Latest Technology in Engineering, Management & Applied Science (IJLTEMAS)* ISSN 2278-2540, New Delhi, India, **V**ol 8, issue 2, pp. 1-5, Feb. 2019.
26. Halstead, M. H.'Elements of Software Science, Operating and Programming Systems Series', ElservierComputer Science Library North Holland N. Y. Elsevier North-Holland, Inc. ISBN , 1977.
27. Halstead, M.H. Elements of Software Science. Elsevier North-Holland, New York, 1977.
28. McCabe, T. 'A Complexity Measure'. IEEE Transactions on Software Engineering, 1: p. 312-327, 1976.
29. McCabe, T.J., Watson, A.H. (2010): Software Complexity, McCabe and Associates, Inc. (last accessed 17.03.2010) Available at: http://www.stsc.hill.af.mil/crosstalk/1994/12/xt94 d12b .asp

.