

ON-LINE HELP

The MetricAdvisor environment has context sensitive on-line help. Users can select help for overall product features or for a specific topic. Within the on-line help, users can navigate by following hyperlinks to related topics. The help gives information to the users that allow them to interpret the metrics calculated.

Other Features

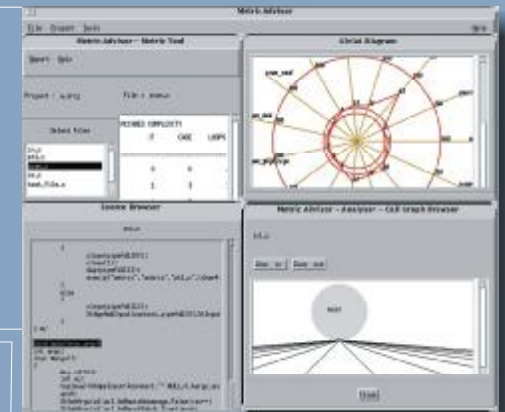
The Metric Advisor allows the user to change the default threshold values and also indicates the extent to which the measured value exceeds the threshold value.

A major area in Software Engineering is the maintenance process. It consumes 40-75% of the software effort. The first step in reducing the maintenance effort is to find the code that is error prone and expensive to maintain.

The use of Metrics helps the developer in identifying source code that is error prone and difficult to test. Metrics also provide an insight into the software, as well as the processes used to develop and maintain it. The Metric Advisor* helps in developing a highly maintainable product by evaluating key software metrics and graphically presenting the structure of the program.

DESCRIPTION

The MetricAdvisor* calculates the following metrics: McCabe metrics that helps in estimating code complexity for identifying the most risky code, Halstead metrics that helps in calculating the programming effort in man-months, Complexity Density Metrics that helps in predicting the maintenance productivity and Fan-in, Fan-out metrics that helps in locating modules that contribute the highest maintenance effort. Besides these, it also helps the user to understand the structure of the code.



Metric Advisor - Overall View

AVAILABILITY

Supported Hardware	: UNIX Workstations
Supported Operating System	: AIX, Solaris and Linux
User Interfaces	: GUI
Supported Languages	: C
Prerequisite Software	: Java

*All trademarks and brand names are owned by their respective owners.

KEY FEATURES

- ▶ Halstead metrics for man-months calculation.
- ▶ Cyclomatic Complexity for identifying the most complex modules.
- ▶ Complexity Density Metrics for maintenance productivity.
- ▶ Fan-In and Fan-out metrics for estimating maintenance complexity.
- ▶ Commentedness and LOC metrics for maintenance and productivity estimation.
- ▶ Call graph for source code analysis.
- ▶ Graphical display using Kiviat and X-Y diagrams.
- ▶ User configurable threshold values for the metrics.
- ▶ Context sensitive help.



Centre for Development of Advanced Computing

C-DAC Knowledge Park, No. 1, Old Madras Road, Byappanahalli, Bangalore - 560 038, India
 Tel: +91-80-534 1874, 534 1909 Fax: +91-80-524 7724
 e-mail: bdm@cdacindia.com website: <http://www.cdacindia.com>

Head Office

Pune University Campus, Ganeshkhind,
 Pune - 411 007, India
 Tel: +91-20-569 4000/01/02/03
 Fax: +91-20-569 4059

New Delhi

A 335, Shivalk Enclave,
 Near Malviya Nagar,
 New Delhi - 110 017
 Tel/Fax: +91-11-687 4689/91/97
 e-mail: bd@cdacindia.com

Hyderabad

2nd Floor, Delta
 Chambers,
 Ammerpet,
 Hyderabad - 500 018
 Tel: +91-40-340 1331/32
 Fax: +91-40-340 1531

• Chennai: +91-44-371 9226/27

• Kolkata: +91-33-321 2357

• Thiruvananthapuram: +91-471-554086

The MetricAdvisor has two components: the **Metric Tool** and the **Analyser**.

Metric Tool

The Metric Tool is the major component that calculates a variety of source code metrics by parsing the source code. It also compares the software against the configured quality threshold. It prints a report of the quality level of the software to help the developer focus on the portions of code that need to be reworked. The user can take corrective measures to make the code meet the expected standards.

The Metric Tool calculates the following metrics.

- Cyclomatic complexity
- Fan-in and Fan-out
- KLOC
- Commentedness
- Complexity Density
- Halstead Software Science Metrics.

ANALYSER

The Analyser helps the developer in locating the various identifiers and understanding the call graph of the code. It parses the 'C' Language source code and lists out the functions, structures, macros, global variables and externals in the programs. It also includes a source browser that allows the user to click on an identifier and view the corresponding source code. The local variables in a function can also be listed.

The call graph supports zoom in and zoom out facilities. The user can view the corresponding source code and the complexity metrics of a function by clicking on any node in the call-graph.

Cyclomatic Complexity

Cyclomatic complexity is based on a program control structure and can be calculated from the number of conditional statements in the source code. Functions with a large number of decision statements are difficult to comprehend and more difficult to test. As a result, functions with a high cyclomatic complexity are more error prone. Thus, this metric allows one to focus on the most complex code, which will result in higher rate of defect removal. Complexity metrics are plotted on Kiviat and X-Y diagrams making it easy to spot the most complex and error prone functions.



Kiviat Diagram

Fan-in and Fan-out

The Fan-in and Fan-out metrics are used to estimate the complexity of maintaining the software.

Fan-out indicates the number of functions a function calls. Modifying a function can result in the functions that are called by the modified function. A maintainer of this module needs to understand many other functions that make maintenance harder and time consuming. Therefore, functions with a large Fan-out are more expensive to maintain.

Fan-in is the count of the number of functions that call a function. A function with a high Fan-in means that many functions use it. It could also mean that the function is implementing a number of functionalities. If the specifications of a function with large Fan-in is changed then all the other functions that use this have to be modified.

Commentedness and KLOC

Commentedness indicates the readability of a program, which is important for maintenance and re-engineering.

Program development and maintenance effort is primarily a function of a program size in kilo lines of code (KLOC). The length hypothesis states that the number of bugs is proportional to the number of machine language statements. KLOC can be used for cost estimation of future projects, comparison of products and for determining productivity. Most of the standard cost estimation models depend on the program size.

Complexity Density

Complexity Density is inversely proportional to the maintenance productivity. Maintenance productivity can be defined as the sum of the total number of lines divided by the time in hours spent on coding and testing upgrades to the software. Thus, Complexity Density also can be used as a measure of the difficulty of maintaining a program.



Call graph

Halstead Software Science Metrics

Halstead metrics is based on the fact that the complexity of a program is related to the number of operators and operands in the program. Operators are syntactic elements such as +, -, <, > and operands are quantities that receive the operators namely variables and constants. Halstead metrics calculates the Program Volume, Difficulty and Programming Effort. The Effort measured in the Halstead metrics is a measure of the effort invested in man-months programming the module.