Chapter Five

JAVA METRICS GENERATOR:  
A NEW METRICS TOOL
CHAPTER 5

JAVA METRICS GENERATOR:
A NEW METRICS TOOL

"Since the measuring device has to be constructed by the observer ... we have to remember that what we observe is not nature in itself, but nature exposed to our method of questioning."

Werner Karl Heisenberg (1901 - 1976) in "Physics and Philosophy" [1958]

5.1 Introduction

The quality of any software measurement program is largely dependent on careful data collection. But collecting data is easier said than done, especially when data are collected across a diverse set of projects. Thus, data collection is becoming a discipline in itself, where specialists work to ensure that measures are defined unambiguously, that the collection is concise and complete, and that data integrity is not at risk. But it is acknowledged that metrics data collection must be planned and executed in a careful and sensitive manner [Fen 96].

The implementation of software metrics collection process manually is a time-consuming and laborious task for software engineers and project managers. For metrics to be actively used they need to be automated, easy-to-use and flexible enough to meet different requirements and goals [Xie 99], [Ban 97]. As object oriented technology is gaining popularity, object oriented
metrics tools also become a popular category of software metrics tools. Tools for object-oriented metrics are essential in real world software development [Ban 97].

The software metrics tools can be divided into three categories [Kuh 94], [Ban 97]:

- **Tool Integrated Facilities:** Some facilities for software measurement have been integrated in some software development tools (e.g., integrated into the programming language). For instance, ObjectWorks/Smalltalk 4.1 compiler is a tool integrated facility. It can measure the number of instance variables, instance methods, and class variables of the class.

- **Tool Extended Facilities:** Users of a software development tool can extend the measurement facilities in the tool. For example, the language Smalltalk IV can be extended to provide system measures such as depth and width of inheritance hierarchy, average number of subclasses, methods and variables.

- **Stand Alone Measurement Tools:** These are software tools specifically developed for performing metrics data gathering, analysis, and visualization. Most of the currently available tools in this category support traditional metrics [Xie 99].
There are various object oriented metrics tools available in the market. Arockiam et al [Aro 03] have done an extensive survey of the popular object oriented metrics tools. Some common object oriented metrics tools include the Java Measurement Tool, Jade Bird Object oriented Measurement Tool, Object oriented Software Measurement Tool, etc. Inspite of the availability of a good number of metrics tools in the literature, many organizations are developing metrics tools suiting their own needs. But there is no general model found in the literature for the development of the metrics tool. In order to aid the developers who intend to develop a new metrics tool, the author has proposed a new model namely Model for Object Oriented Metrics Tool (MOOMT). Using this model, a new tool namely Java Metrics Generator (JMG) has been developed for the generation of metrics values from Java projects. The proposed model MOOMT and the metrics tool JMG are described in this Chapter.

5.2. Model for Object Oriented Metrics Tool

A model is an abstraction of real world objects, process etc. representing only the essential characteristics. Models are used to understand something before it is being built. The Model for Object Oriented Metrics Tool, which is proposed by the author, is a generalized notion that can be elaborated on to
develop a tool to suit the user’s needs. Fig. 5.1 gives the diagrammatic representation of the tool model.

Fig. 5.1 Object Oriented Metrics Tool Model

The success of implementation of any model depends on the degree of completeness of the model. Of course, the ability to represent the real world object or process as a model depends on one’s own knowledge and experience. Further, for the effective design of the model the basic requirements should be

---

¹ This model has been presented as a paper titled, “A Model for Object oriented Metrics Tool”, Proceedings of National Seminar on emerging trends in high performance computing and communication, Department of Information Technology, KSR College of Arts and Science, Tiruchencode, August 2005.
clear. In designing and implementing a metrics tool model, the following set of questions may help the designer to ensure a planned development leading to a useful and interactive tool.

1. At what levels of the program metrics are required?
2. What aspect of the program needs to be measured?
3. What metrics can be accurately collected?
4. Which of the metrics can be directly generated and which have to be calculated?
4. Is the formula for metric to be calculated known?
5. Will the metric collected be beneficial in assessing the program?
6. Is the impact of metric value on quality of the product known?

If the above mentioned questions are properly answered in the requirements, the tool developed will be effective.

The proposed model for development of object oriented metrics tool comprises of the following modules:

1. Input Module
2. Metrics Generation Module
3. Metrics Storage Module
4. Metrics Calculation Module
5. Quality Interpreter Module
6. Report Generation Module
The different modules in the tool model are explained in detail in the following sub sections.

5.2.1 Input Module

The input module of the model enables the user to input the class or file for the collection of metrics data. It can be made interactive and user friendly by using the Graphical User Interface tools. For this purpose a browser window may be designed so that the user can select the file that he wants to analyze from the browse list. Appropriate feedback is to be given to the user on selection of the file. Provisions should be given to the user to perform the analysis at the method level or at the class level or at the system level.

5.2.2 Metrics Generation Module

The metrics generation module has two sub modules namely metrics definition module and metrics generator module which are described in the following sub sections.

5.2.2.1 Metrics Definition Module

Metrics Definition involves defining the set of metrics that are to be automated for measurement and quality assessment. The tool developer should analyze the requirements and make a list of all metrics that he wants to collect. These metrics may be classified as size, reuse and complexity metrics. Depending on the granularity, the metrics may also be classified into different levels viz. variable level, method level, class level or system level.
Once the list of metrics to be collected is ready, they should be defined. One may adopt the list of metrics and their definitions proposed by either Chidamber et al. [Chi 94] or MOOD metrics by Brito e Abreu [Bri 94], or by Lorenz and Kidd [Lor 94] or by Li and Henry [Li 93] or a new set of metrics defined by the organization suiting to their needs or a combination of these. For instance, the simple Line of Code metric means differently to different collectors. The exact definition, as applied in the counting, should be stated. Appropriate parsing algorithm has to be designed and implemented to generate the required metrics using the definitions. These definitions should be stated in the manual of the tool as well so that the user knows what the metric implies.

5.2.2.2 Metrics Parser and Generator Module

The Parser sub module should be capable of identifying the valid token of the language for the defined set of metrics. Efficient parsing algorithms should be implemented in the design of the parser. The Metrics Generator is a part of the tool which generates the metrics that are predefined in the Metrics Definition stage. Language characteristics are to be considered while parsing the input code for the metrics. The input file is parsed for all keywords and punctuations specific to the language used in developing the software under analysis. The generator module should be designed with the provision to add new parser sub modules for a specific language. The generated metrics should
be stored in the metrics database / file classified according to the method, class
and file.

5.2.3 Metrics Storage

The metrics generated directly from the code are called direct metrics. The metrics such as comment percentage, coupling factor etc. are calculated from the direct metrics and hence they are called derived metrics. The metrics calculator module should be designed with appropriate computation definitions for generating the derived metrics from the metrics storage.

The direct as well as the derived metrics should be stored in a database / file and can be used in the future for analysis and quality interpretation. The metrics store should be designed with suitable data structure and it should be evolvable. The database should be capable of maintaining a history of metrics data over various versions during evolution. It should also be capable of providing statistical information on the metrics data. Security has to be ensured to the database so that only the authorized users will access the metrics data for any analysis.

5.2.4 Metrics Calculator Module

Metrics Calculator takes up the generated metrics and computes those metrics, which are not directly obtainable from source code. In most cases, the calculated metrics include the averages, percentage, minimum, maximum, ratios, index values etc. The calculated metrics allow the user to analyze the
products or files as a whole and study the general characteristics of the design of the software. It helps in making intra module and inter module comparisons. The high level statistical information can be used for understanding the difference in design characteristics of two different development environments.

5.2.5 Quality Interpretation Module

The parameters for understanding the quality of design may be defined for every development environment. The predefined thresholds can be stored in a separate database and can be compared with the actual metrics data. Thresholds are defined as "heuristic values used to set ranges of desirable and undesirable metric values for measured software. These thresholds are used to identify anomalies, which may or may not be an actual problem [Lor 94]". The thresholds can be obtained from the previous studies or experience. The thresholds can also be stored in a separate database and compared with the values in the metrics database. Different colors can be used to alarm the metrics data that are above the thresholds. The Quality Interpreter does this job and suggests whether the file being analyzed is in step with the proper values.

5.2.6 Report Generator

Report generation involves presentation of the metrics data, summary and the associated data on quality of design to the analyzer in a suitable format. The Report Generator component should provide the analyzer with tabular, graphical and textual format of the output. The output if required, should also
be exportable as text document or in a suitable format for a spreadsheet or in a format suitable for statistical packages for further analysis.

5.3 Java Metrics Generator: A New Metrics Tool

The new tool, namely Java Metrics Generator (JMG), has been developed based on the proposed model presented in the subsection 5.2 with all basic functional modules like parser, metric generator, metric calculator etc. The specific feature of the model is that it can be updated to add other analyzer modules or other parser modules along with the existing analyzer and parser modules, to enable analysis of source codes of different languages in the future. The architecture of the new tool is shown in Fig. 5.2.

![Diagram of Java Metrics Generator architecture](image)

**Report Generator and output**

*Fig. 5.2: Architecture of Java Metrics Generator*
The newly developed object oriented metrics tool consists of five modules.

- Input module
- Parser and Metrics Generator module
- Metrics calculator module
- Metrics Store module
- Report generator and output module

These modules are arranged in a multistage graph approach. A multistage graph is a graph with a head and tail node where there exists more than one path between the head and tail node. In this new model the input module (front end) and output module (display module) can be considered as the head and tail nodes respectively. The user of this tool can make his own choice to travel in any path between the input and output module. In the following subsections, all these modules are described in detail.

5.3.1 Input Module

The input module is otherwise called the front-end module. This module enables the user to have a direct interaction with the tool. The user can enter the input via this module. The input to the module is the name of the source file that is to be analyzed. The user can also choose any one of the two available modes of execution viz.
1. Window based execution

2. Command line based execution

The window based execution will initiate operation in a GUI based environment in which the user can easily select and analyze the source file. But in the command line based execution, the user has to type the input file name along with the complete path. Once the type of execution is chosen the user has to use the same environment throughout the processing.

5.3.1.1 Window Based Execution

The behavioral modes of the application are completely hidden from the user. The procedure to be followed by the user for analysis of Java files are briefly summarized below:

- The user can enter the input i.e. the file name, in the text box if he has to process a single file.
- The user can click the open button to get a file dialog box to select a particular directory where all the sources files of a project are available.
- The user can choose list button if he needs to analyze different files from different projects.
- Another option which has been included in this application, but not available in any other tool, is that the user is asked a question whether he wants to analyze the complete file or just
Similarly for choosing the output, two parameters should be entered.

First parameter is the file name and second parameter is the format of output required (table/text).

The only disadvantage with command line execution is that a list of files can be executed only by entering the complete path for each file.

5.3.2 Parser and Metrics Generator Module

The parser module is considered to be the heart of the tool. The parser module breaks the given source file into smaller individual tokens and sends these tokens to the Metrics Generator modules for analysis. The parser module normally accepts the textual form of the input file. If the input file is a compiled code then the parser disassembles the code.

The tokens are checked for keywords such as "class", "public", "private", "protected", etc for identification of class and the access specifiers. For variable declarations, the keywords checked are "int", "string", etc. For checking inheritance, the keyword "extends" or "implements" is searched in the tokens. If the base class method and the derived class methods share the same name, it is counted as an overridden method.

The Metrics Generator module analyzes the tokens and generates the metrics. This module generates direct metrics on size, complexity and reuse
aspects. The following are the direct metrics generated from the output of the parser.

**Size Metrics**

- Total lines of code in each file.
- Total lines of code in each class of a particular file.
- Total comment lines in each file.
- Total multi-line comments in each file.
- Total blank lines in each file.
- Total number of methods in the file.
- Total number of public methods in the class.
- Total number of imported packages in each file.
- Total number of variables in each file.
- Total number of private variables in each file.

**Complexity Metrics**

- Weighted Method per Class (WMC)
- Weighted Method Complexity per Class (WMCL)
- Coupling Between Objects (CBO).
- Response For a Class (RFC).
- Lack of Cohesion Of Methods (LCOM).
Reuse Metrics

- Depth of Inheritance Tree (DIT)
- Number Of Children (NOC)
- Total Number of Methods Overridden (NMO)
- Total Number of Methods Inherited (NMI)
- Total Number of Attributes Inherited (NAI)

5.3.3 Metrics Calculator Module

The task of the metrics calculator is to compute the derived metrics for size, complexity and reuse from the generated direct metrics. The main objective of derived metrics is to identify potential problems in the early stages of the development process. This will help to create awareness regarding the quality issues such as reliability, testability and maintainability. The derived metrics that are calculated are listed below:

Derived Size Metrics

- Average number of classes in the system.
- Average number of methods in the system.
- Average LOC per class.

Derived Complexity Metrics

- Average method complexity.
- Coupling factor.
Derived Reuse Metrics

- Average depth of inheritance.
- Average number of children.
- Average number of methods overridden.
- Average number of overloaded methods.
- Method inheritance factor.
- Attribute inheritance factor.
- Specialization Index.

5.3.4 Metrics Store Module

The direct metrics values generated by the Metrics Generator are stored into a file. Metrics calculator module reads these direct metrics data and calculates the derived metrics and they are stored into another file.

5.3.4 Report Generator and Output Module

The report generator module reads the metrics data from the metrics store (direct metrics file and derived metrics file) and produces the reports of the metrics collected and calculated. Two types of reports are generated for each file.

They are

1. Class level analysis report
2. Overall analysis report
These reports are the summarized forms of metrics data collected.

Both in window based and command line based mode of operation, user is asked to select his display format (either textual or tabular). The user can view the results of his analysis in class level, method level and system level.

There are two types of display formats

1. Table format.

2. Text format

The table format displays the results in rows and columns. This display would be simple to understand if the results are required at in class or method level. The text format will be easy to understand by the users in all levels.

5.4 Operational List

1. Java Metrics Generator works on a flow graph based paradigm as depicted in Fig. 5.2. The user may traverse through any path in the graph. It allows the user to analyse either the entire project or selected files from the project. He may choose either GUI or command line environment to operate.

2. The project for analysis can be chosen either by browsing through the directories or by typing the full project name in the text box.

3. After entering a single file or choosing the directory containing the project files for analysis the file that are to be analyzed by the parser are displayed in a list box for verification of the user.
4. After the project has been selected, the user will proceed to select type of analysis (size or reuse or complexity or all the three).

5. Three list box are displayed on the screen as default output of analysis. The first one consists of the file(s) in the project given for analysis. The second lists out the classes in files. The third one displays the metrics at class at method level.

6. The user is allowed to choose the type of output format (either tabular or textual) by clicking on the radio buttons that are displayed.

7. If tabular format of output is chosen the user will be given an option to view it class wise or as summary for all classes. The summary for all classes gives the system / project level analysis report.

8. For every file analyzed, the corresponding metrics data are stored in a file having the same name that of the file being analyzed. If text is selected as the output option by clicking the “open” button the user can open the output file. Then, a file dialog box appears from where the user can select the files and view its content.
5.5 Conclusion

The newly developed tool is user-friendly as it is designed for both GUI based environment and command line based environment. The tool can be updated in future to analyze projects developed using the new object oriented languages by adding the appropriate language parser and other report generation modules. The tool has been successfully tested using sample projects and the results were verified manually. The screen shots of the sample runs are given in Appendix B.