JBOOMT: Jade Bird Object-Oriented Metrics Tool[*]

Tao XIE, Wanghong YUAN, Hong MEI, Fuqing YANG

(*Department of Computer Science & Technology, Peking University, Beijing 100871*)

Abstract: Focusing on software productivity and software quality control has spurred the research on software metrics technology. The increasing importance being placed on object-oriented software development has led to the research on the object-oriented software metrics and the development of automated tools to support object-oriented metrics. To effectively aid the software evaluation, a software metrics tool is supposed to support the metrics model. The objective of Jade Bird Object-Oriented Metrics Tool (JBOOMT) is to provide an automated software metrics support for users and managers to measure the design or source code of the object-oriented program and thus evaluate the quality of the software according to the specified hierarchical metrics model. A mechanism is provided for metrics users to customize the preferred metrics model and browse the details of the metrics model. This article introduces the design of JBOOMT and discusses its implementation built in the Jade Bird Program Analysis System (JBPAS).

Keyword software metrics, object-oriented metrics, reusability metrics

1. Introduction

Measurements have been widely used in many engineering disciplines, yet in the computer software industry there still have been some doubts about their use in this field. During recent years, the interests in software metrics have grown both greatly and steadily in software industry. With the project programmer and manager focusing on software productivity and software quality, there exist needs for better technique of software development and software metrics during the process of development. Recently object-oriented technology is becoming increasingly popular in industrial software development environments. This technology offers support to provide software product with higher quality and lower maintenance costs. Since the traditional software metrics aims at the procedure-oriented software development and it can not fulfil the requirement of the object-oriented software, a set of new software metrics adapted for the characteristics of object technology is greatly in want. Accordingly object-oriented metrics then becomes an essential part of object technology as well as good software engineering.

To apply the object-oriented metrics technology in practice effectively, an automated object-oriented metrics tool is necessary to support the activities during the process of software evaluation. Currently there already exist some metrics tools to aim at this objective. However, most of them only produce certain unstructured independent metrics results. To attain some meaningful metrics data in customer and management level, it is required to formulate metrics models to incorporate the related metrics categories into an integrated framework. An object-oriented metrics tool to support such models is greatly desired. Jade Bird Object Oriented Metrics Tool (JBOOMT) is designed to measure the design or source code of the object-oriented program automatically and thus evaluate the quality of the software according to the specified

[*] Sponsored by State 9th Five-Year Plan, State 863 high technology program, and Ricoh Company, Ltd., Japan.

hierarchical metrics model. A mechanism is provided for metrics users to customize the preferred metrics model and browse the details of the metrics model by using a metrics model database in the tool.

This paper describes our research and experiences in developing JBOOMT. It is organized as follows. First, section 2 details the design of JBOOMT, which comprises three modules: analyzing, calculating and displaying. Section 3 discusses the Jade Bird Program Analysis System (JBPAS) which JBOOMT is built in and the application of JBOOMT in component-based software development environment briefly. Section 4 discusses some related research work. Finally, section 5 gives the conclusion.

2. Jade Bird Object-Oriented Metrics Tool

Jade Bird Object-Oriented Metrics Tool (JBOOMT) is designed and developed to support object-oriented software quality assessment. The tool makes it easy for the hierarchical model to be defined or constructed, and put into practice.

After metrics user customizes the metrics model whose lower level metrics can be selected from a list of implemented metrics, user can store its specification in the model database. According to the specified model, the tool calculates the related program information and derives the values of the model from the information database which contains comprehensive information of the program, which are then stored into metrics result database. Figure 1 shows the architecture of JBOOMT.

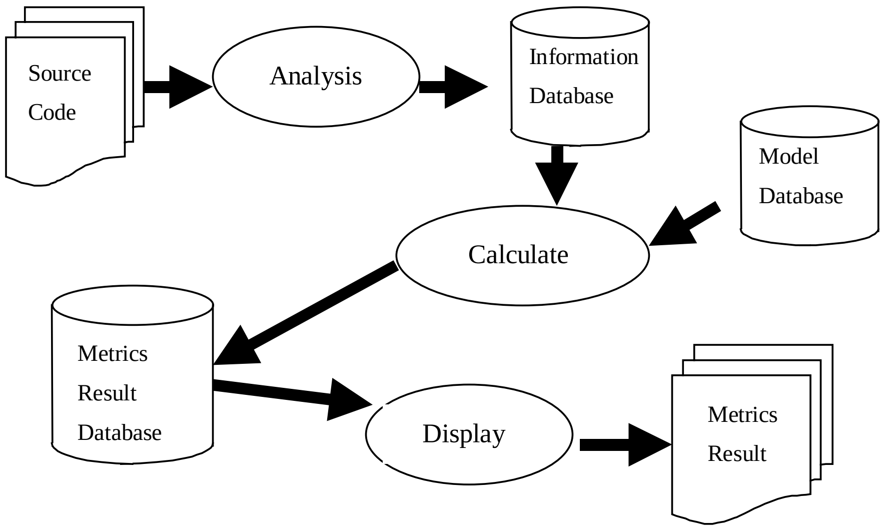


Figure 1. The architecture of JBOOMT

As shown in figure 1, the metrics measurement consists of three phases: analyzing, calculating and displaying. According to these three phases, the tool is divided into three corresponding modules.

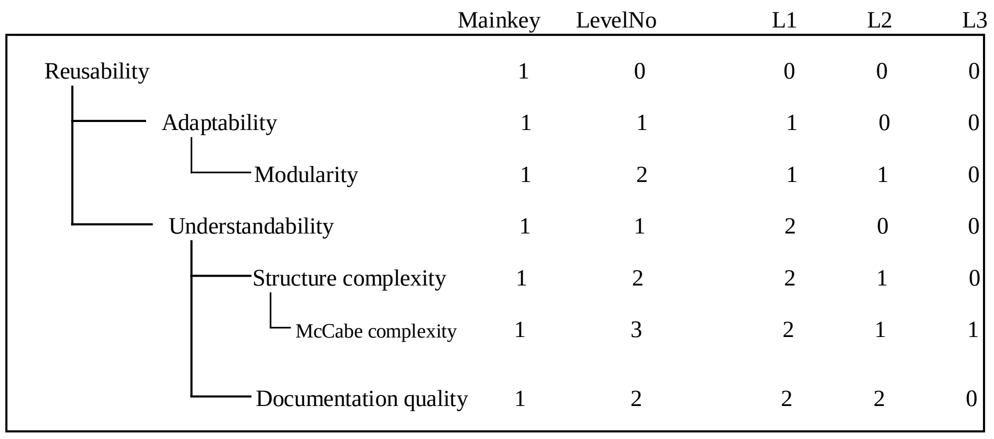
In the analyzing phase, the analysis front end analyzes source code, extracts program information and stores it into the program information database through the database server 〖1〗. A front end is developed to analyze the source code syntactically and semantically which includes both the interface and implementation part of the program.

In the calculating phase, according to the selected model in the model database which have already been customized by metrics user, all the values of the model are calculated from the information database and then are stored into metrics result database. The model database is used to store the definition of some hierarchical models and the metrics result database is to store the result values of the calculated hierarchical model.

The metrics models are classified as method, class and system level metrics model based on the scope of the measured object. When one specific method of a class is assessed by calculating the metrics data from the information within one method of the class, the corresponding model is classified as method metrics model. If one class is evaluated, the required information for obtaining the metrics data is often limited within the scope of one class, then the model is classified as class metrics model. Similarly in order to evaluate the whole software system, the scope of the information extends to more than one class, and this metric model is called system metrics model. Each model is organized as a hierarchical diagram. The value of upper level node is calculated based on the lower level node.

In some popular models, items in the model except for the items of the bottom metrics level can generally be calculated from the values of lower level items according to the corresponding weight value assigned to them. The values of the bottom metrics level can be directly calculated from the source code information. The value of each item in the model should be normalized so that it yields a value between 0 to 1. A value close to 0 indicates that the measured characteristic may cause problems, while a value close to 1 indicates that the corresponding characteristic is kept inside its limits.

Figure 2 shows the representation of hierarchical structure in model database and metrics result database. It defines the structure information of the hierarchical model.



	Mainkey	LevelNo	L1	L2	L3
Reusability	1	0	0	0	0
Adaptability	1	1	1	0	0
Modularity	1	2	1	1	0
Understandability	1	1	2	0	0
Structure complexity	1	2	2	1	0
McCabe complexity	1	3	2	1	1
Documentation quality	1	2	2	2	0

Figure 2. The representation of tree structure of model in relational database

To simplify the explanation, the representation of a 4-level model is illustrated in figure 2. On the left part of the figure there is a sample metrics model in which the nodes' structure information is stored in a relational database with the format in the right section of the figure. The hierarchical structure is specified by determining the values of Mainkey, LevelNo, L1, L2 and L3. Mainkey is the unique identifier of the tree and the nodes from different trees has different Mainkey but all nodes of the same tree have same Mainkey value. LevelNo represents the number of the level in which the node is located, which is counted beginning from 0 (root node level). For example, the node "Adaptability", whose LevelNo value is 1, is located at 1 level of the tree.

Sequence number of a node is defined as the order (from right to left, and from top to bottom showed in figure 2) of this node in the children list of its parent node. For example, the sequence number of node “Understandability” is 2 and “Structure complexity” is 1. Li of a node represents the sequence number of its i-level ancestor node. And when this node has no i-level ancestor node, if this node itself is at i level, then Li is the sequence number of this node, and otherwise Li is set as 0. For example, “McCabe Complexity” node has 3 ancestor nodes: “Reusability” at 0 level, “Understandability” at 1 level and “Structure complexity” at 2 level, therefore, “McCabe Complexity” node’s values of L1, L2 and L3 are 2, 1 and 1 respectively.

When a 6-level is represented, it also required the value of L4 and L5. In fact, the value of L4 and L5 is set to 0 in the example showed in figure 2. The values from L1 to L5 is reserved in the database of JBOOMT which means JBOOMT support at most 6 levels metrics model for in the real world the metrics models are generally less than 6 levels.

The nodes in the model tree can be classified into two types: internal node and external node. External node has no offspring nodes and internal node has at least one offspring node. The value of each internal node can be calculated from weighted children nodes which are associated with specific weight values. As for each external node, its value can be obtained directly from source code by means defined in metric type table.

As proposed by Karlsson 〖2〗, All metrics should be of one of the following types, and each type of metrics should be normalized to yield values ranging between zero and one according to following specified formula.

1). Upper limit metric which is characterized by a break-off value *a* and a 50% limit *b*. When the measured value *m* reach the break-off value *a*, the normalized value *M* will start to decrease. For some instances in point, the inheritance depth of class and McCabe’s cyclomatic number fall into this type. From the qualitative perspective, the lower the measured value is, the better the quality of the evaluated object is and it means that the normalized value is much closer to the value 1. Figure 3 shows the relation between the computed metric *M* and the measured software measurement *m*.

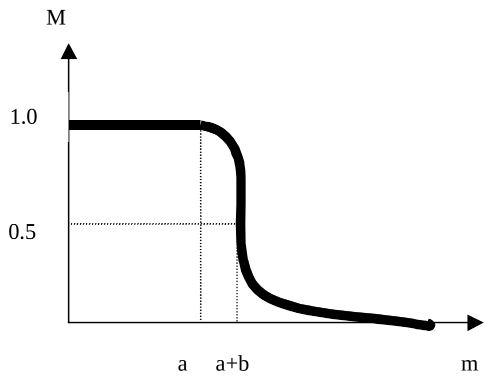


Figure. 3 Upper limit metric 〖2〗

The formula for this type of metric is described as:

$$M(m) = \frac{1}{1 + e^{(x \times (m - y))}}$$

In this formula, the parameters *x* and *y* can be calculated from these two relations: *M(a)*= 0.99 and *M(a+b)* = 0.5. In this case, it is necessary to demarcate the thresholds of *a* and *b* according to the specific metric.

2). Optimum value metric which is characterized by center value *a* and 50% limits at *a+b* and *a-b*. Qualitatively speaking, when the measured value is too small or too large, the quality of the considered object is not good enough. The measured value around *a* is viewed as the ideal value. Some examples in this case are the average member function size measured by LOC per member function and relative number of comments in a method. Figure 4 illustrates the relation between the computed metric *M* and the measured software measurement *m*.

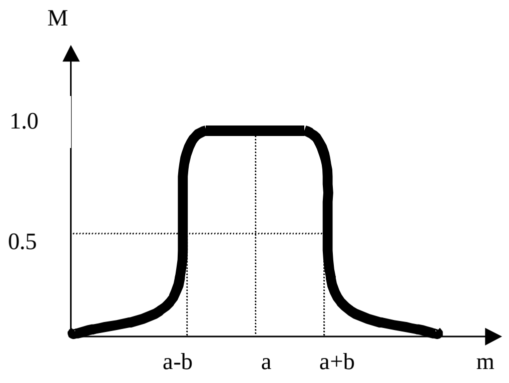


Figure 4. Optimum value metric 〖2〗

The formula for this second type of metric is defined as:

$$M(m) = e^{-x \times (m-a)^{z}}$$

The parameter *z* specifies the squareness of the graph. And parameter *x* also can be calculated from these two relations: *M(a+b)=0.5* and *M(a-b) = 0.5*. Similarly the thresholds of *a* and *b* also should be demarcated according to the specific metric.

3). Linear dependency metric which is characterized by the intercept *b* of the measured value *m* coordinate. It is a simpler type of metric whose typical examples are the inheritance generality and the relative number of system-dependent code lines. The formula for this type of metrics is simply defined as M(m)= 1-m/b. Figure 5 illustrates the relation between the computed metric *M* and the measured software measurement *m*.

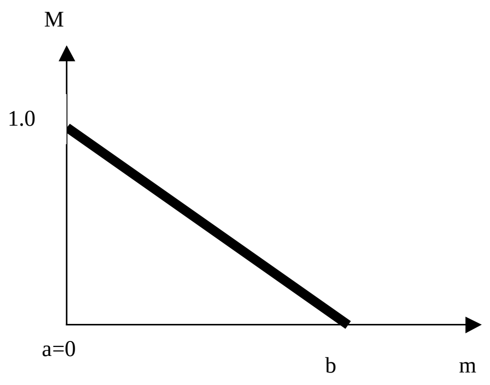
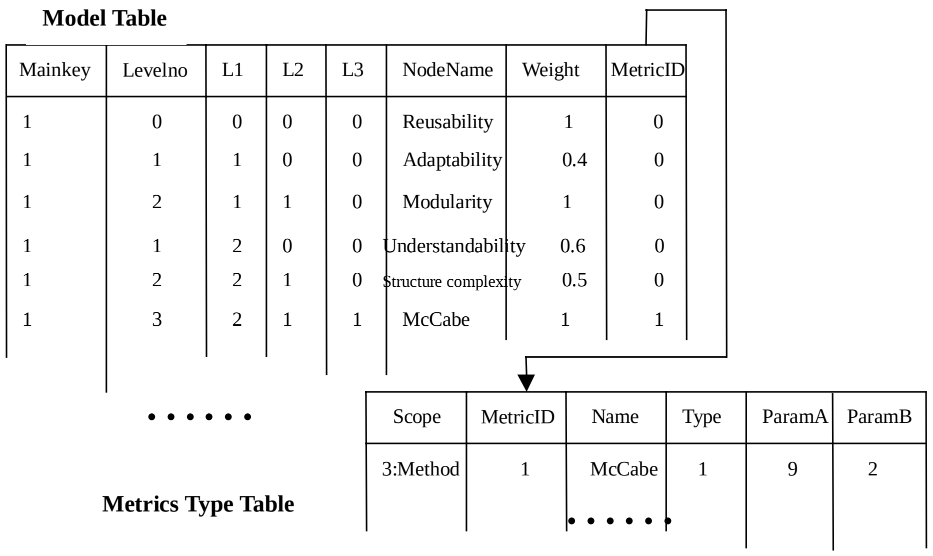


Figure 5. Linear dependency metric

The information of the available metrics is stored in metric type table of model database. According to the three kinds of metrics model, there are also three corresponding scopes of metrics. System scope of metrics mainly deals with the metrics of the project scope such as total number of files or modules, functions, macros, classes, source lines of code, and average derived classes per class, average parameters per function etc. Class scope of metrics generally comprises the metrics of a class like Weighted Methods per Class (WMC), Depth of Inheritance Tree of a class (DIT), Number Of Children of a Class (NOC) and Response For a Class (RFC) etc. 〖3〗. Additionally method scope of metrics consists of the metrics of a method in a class such as number of messages sent of a method, lines of code of a method etc.

The metric scope and parameters for normalization such as the metric type, "*a*" value and "*b*" value should be stored in the metrics type table. The information about the hierarchical structure and the associated weight value should be stored in the model table and if node type is a measurable metrics type, the related metrics ID from metric type table also should be specified in model table. The structure of the tables in model database is showed in figure 6.



Model Table

Mainkey	Levelno	L1	L2	L3	NodeName	Weight	MetricID
1	0	0	0	0	Reusability	1	0
1	1	1	0	0	Adaptability	0.4	0
1	2	1	1	0	Modularity	1	0
1	1	2	0	0	Understandability	0.6	0
1	2	2	1	0	Structure complexity	0.5	0
1	3	2	1	1	McCabe	1	1

• • • • • •

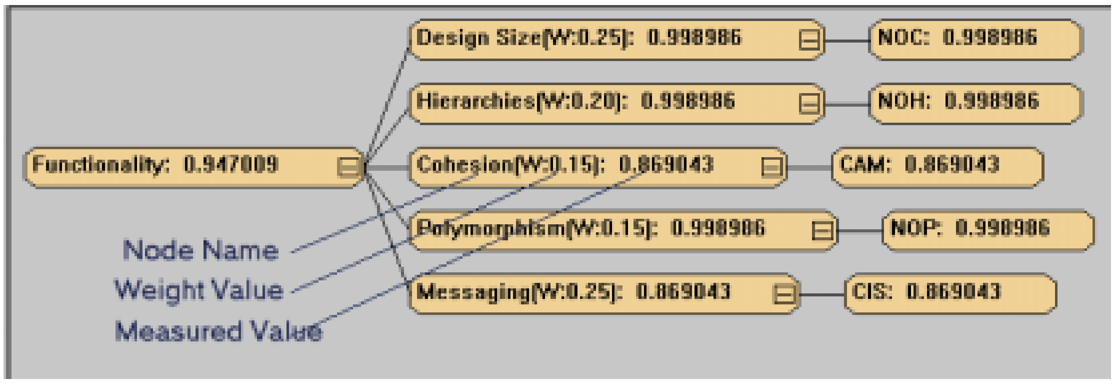
Metrics Type Table

Scope	MetricID	Name	Type	ParamA	ParamB
3:Method	1	McCabe	1	9	2
		• • • • • •			

Figure 6. Structure of tables in model database

After metrics user selects the model, which is required to calculate, the tool statistically calculates the value from the raw information database using SQL database query language according to the specification of the metrics model customized by the metrics user. Because the information database contains complete information of the source code, it is satisfactory enough to derive desirable metrics results from the information database. When the measured data is available, the normalized value ranging between 0 to 1 can be calculated by the way defined in the metrics type table. Finally, the derived metrics result can be stored in the metrics result database for later use, whose structure is similar with the model database.

In the displaying phase, The interface component loads the metrics data from metrics database and provides visual presentation such as chart, graph or illustration to display the metrics results. The values of each node in the hierarchical model can be displayed to metrics user. At same time user can easily tailor the thresholds and default values of the metrics through the graphic user interface. The tool presents the result from different perspectives to facilitate program assessment. Figure 7 show the representation of a metrics model in the tool.


Functionality: 0.947009
Design Size[W:0.25]: 0.998986
Hierarchies[W:0.20]: 0.998986
Cohesion[W:0.15]: 0.869043
Polymorphism[W:0.15]: 0.998986
Messaging[W:0.25]: 0.869043
NOC: 0.998986
NOH: 0.998986
CAM: 0.869043
NOP: 0.998986
CIS: 0.869043
Node Name
Weight Value
Measured Value
Figure 7. An example of metrics model showed in JBOOMT

3. Jade Bird Program Analysis System

The Jade Bird project is a key science and technology project supported by government and has gone through the Sixth, Seventh and Eighth national “Five-Year Plans”. The goal of Jade Bird project is to establish the fundament for Chinese software industry, popularize software industrialization technology and mode, and provide necessary instrument and equipment of industrialized production to Chinese software enterprises. The research and development of Jade Bird III System is a key part of Jade Bird project during the national “Ninth Five-Year Plan”. The Jade Bird III system supports software industrialization production based on component-architecture model 〖4〗.

JBPAS (Jade Bird Program Analysis System) is a program understanding system for C++ programs, which is an important part of Jade Bird System III. It consists of three major components: a front end, a database server and a set of analysis tools. Its architecture is shown in figure 8. The essential part of the information extractor is the C++ front end, which analyzes C++ source code by means of incremental parsing, extracts program information and stores it into the program information database. There are several code analysis tools based on the front end and more are anticipated to appear in the future. JBPAS is to support research in software maintenance, software reuse, reverse engineering, software metrics, and so on. The front end parses source code statically and extracts program information according to the conceptual model into incremental database. Finally, all incremental databases are linked to construct the information database. JBPAS employs EER (Enhanced Entity-Relationship) model to form C++ programs’ concept model, which is relatively comprehensive to support many requirements of program information. 〖1〗.

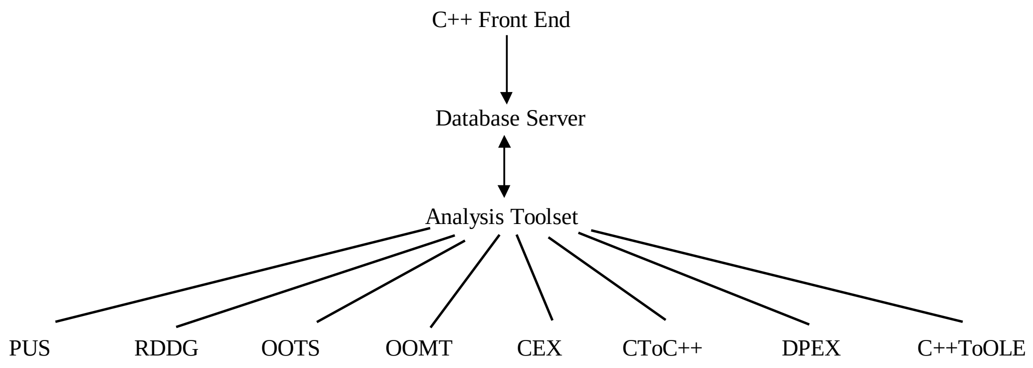


Figure 8. JBPAS Architecture

The toolkit of the JBPAS includes Program Understanding System (PUS), Reverse Design Document Generator (RDDG), Object-Oriented Test Supporter (OOTS), Object-Oriented Metrics Tool (OOMT), Component Extractor (CEX), C To C++ Translator (CToC++), Design Pattern Extractor (DPEX), and C++ To OLE Translator (C++ToOLE). All the above analysis tools share the same program information database through the information manager. And the information manager provides database access interface for other tools, which facilitates the integration of external analysis tools.

JBOOMT is one of the tools in the toolkit of JBPAS and it has been built in JBPAS. JBOOMT has been applied in the component-based software development environment together with other tools in JBPAS. JBOOMT together with Jade Bird Program Understanding System can be used to aid user to understand the legacy systems. JBOOMT can also assist to extract reusable component from existing system. A reusability metrics model can be formulated and then supported by JBOOMT. A set of candidate reusable components is identified by applying this metrics model. Among this set of candidate components, users can use Jade Bird Reverse Engineering Tool (JBRET) to help to reengineer the components. After the adaptation and improvement on the selected components in the reengineering process, components are qualified before inserting into the component library.

4. Related work

In this section we will discuss some related work in the field of software metrics tools.

PROMIS (Program Metric Support) 〖5〗, an automatic software analysis tools generator, is implemented by some extensions of a conventional compiler-compilers. It enables one to describe the desired software analysis tool and its underlying software metrics by applying an extended version of a conventional formal specification language. The specification is used as input and then PROMIS generate a software analysis tool written in a specific programming language by using LEX and YACC. PROMIS mainly deals with some simple conventional programming languages such as C language, but it doesn't seem to be practical to analyze some object-oriented programming languages such as C++, which have rather complex syntax.

OOMetric 〖6〗 is an object-oriented metrics tool for Smalltalk and C++. It can collect some OO-specific metrics data according to class level metrics and method level metrics. It also provides a user interface to allow user to tailor the thresholds of the metrics. The means by which the tool displays its metrics results is to export them to some industry-standard formats such as PDF and RTF. But it doesn't provide enough support for user to evaluate the integral attribute in higher assessment level which metrics users mostly concern with. It only gives metrics user a list of basic unstructured metrics data.

QMOOD++ (Quality Model for Object-Oriented Designs represented in C++) metrics and analysis tool 〖7〗, is a software program automates the process of design selection, design metrics data collection, visualization of design structure, and display of results. The QMOOD relationships, weights, and the equations for computation of quality attributes are implemented through a spreadsheet program, which is easy to modify. The design metrics used to assess design properties are specified in spreadsheet program and can be changed to use data from different metrics. This tool supports the QMOOD quality model well enough, but it seems not to incorporate other hierarchical metrics models easily. And additionally QMOOD++ only collects metric data from the design information, but doesn't analyze the information from the implementation of classes.

5. Conclusion

Currently the hierarchical metrics model is effectively and practically applied in the evaluation of the object-oriented software, so it is significantly important to implement an automated tool to assist the process of evaluation based on the specific hierarchical model. Although there exist many object-oriented metrics tools to assist the software measurement in software industry, most of them simply collect unstructured metrics values, but don't provide a well-defined approach to relate these metrics to the external quality attributes of the software, which is most concerned by metrics users.

To meet this objective, an automated object-oriented metrics tool, JBOOMT, is designed and implemented to support hierarchical metrics models for object-oriented software quality assessment, which can be easily customized by metrics user. Moreover this tool applies the analysis frond-end to analyze not only the design information of the source code, but also the implementation information. For this reason the tool can get complete information of the source code stored in the information database, and this makes it feasible to add many more new metrics based on the information database. JBOOMT has been applied in the component-based software development environment assisting the component metrics, reusable component extraction in legacy system etc.

Currently a front-end for C++ programming languages has already been developed. Because the analysis component and the metrics calculating component is relatively independent by interacting with each other through the information database, it is easy to extend the tool to support other object-oriented programming language by replacing the current front-end with the new one.

References

[1] Fuqing Yang, Hong Mei, Wanghong Yuan, Qiong Wu, and Yao Guo. Experiences Writing C++ Compiler Front End. ACM SIGPLAN Notices, 1998, 33(9): 95-102

[2] Even-Andre Karlsson. Chichester. Software Reuse: A Holistic Approach -Measuring the Effect of Reuse Chapter. New York: Wiley, 1995: 113-180

[3] S. R. Chidamber and C. F. Kemerer. A Metrics Suite for Object Oriented Design. IEEE Trans. Software Eng., 1994, 20(6): 476-492

[4] Fuqing Yang, Hong Mei, Keqin Li, Wanghong Yuan, Qiong Wu. Jade Bird III: A System Supporting Component Reuse. Computer Science, 1999, 26(5): 50-55

[5] Peter Kokol, Viljem Zumer, Janez Brest, Marjan Mernik. PROMIS: A Software Metrics Tool Generator. ACM SIGPLAN Notices, 1995, 30(5): 37-42

[6] Mark Lorenze, Jeff Kidd. Obejct-Oriented Software Metrics. Englewood: Prentice Hall, 1994

[7] J. Bansiya. A Hierarchical Model For Quality Assessment Of Object-Oriented Designs. [Ph.D. Dissertation].Huntsville:University of Alabama in Huntsville, 1997.

Tao XIE M. S. student in computer science at Peking University. He received the B.S. degree in computer science in 1997 from Fudan University. His current research interests are in software reuse, software metrics and program analysis.

Wanghong YUAN M. S. student in computer science at Peking University. He received the B.S. degree in computer science in 1996 from Peking University. His current research interests are in software reuse and object-oriented technology.