International Journal of Computer Theory and Engineering, Vol. 2, No. 5, October, 2010
1793-8201

Quality Metrics Tool for Object Oriented Programming

Mythili Thirugnanam* and Swathi.J.N.

Abstract—**Metrics measure certain properties of a software system by mapping them to numbers (or to other symbols) according to well-defined, objective measurement rules. Design Metrics are measurements of the static state of the project's design and also used for assessing the size and in some cases the quality and complexity of software. Assessing the Object Oriented Design (OOD) metrics is to predict potentially fault-prone classes and components in advance as quality indicators. To perform the assessment accurately, a sequential life cycle model and a well-known OO analysis/design method for java programming language is used. Design metrics helps to identify potential problems in the early stages of the development process. The quality metrics tool has been developed to determine the various design metrics and the quality attributes of Object Oriented program. These quality attributes determines the complexity and efficiency of the program.**

Index Terms—**Metrics, Object Oriented Program, Quality, Design Metrics.**

I. Introduction

Object Oriented Design and development are popular concepts in today's software development environment and in solving software problems. In reality, Object Oriented development has proved its value for systems that must be maintained and modified. The concepts of software metrics are well established, and many metrics relating to product quality have been developed and used. With object-oriented analysis and design methodologies gaining popularity, it is time to start investigating object-oriented metrics with respect to software quality. Producing a high quality software system is a goal required by all the stakeholders involved in any software system project. A major prerequisite to attain high quality software is the continuous and early assessment of its quality. Assessing the quality in the early stages of its development life cycle signals any omissions or weaknesses in quality that need to be addressed, and allows taking the essential measures to deviate such deficiencies as early as possible, thus, reducing the overall cost of software development process, and increasing the software quality level. The difficulty is that, depending on design practices used in a development environment, two metrics may be redundant in one software system, but not in another. There is no such thing as a canonical set of non-redundant metrics that captures all important design properties and is valid for all systems. Therefore, Software Design Metrics features a rich set of metrics, to minimize the risk of missing important design aspects, at the cost of some redundancy among the metrics. There are many tools available but there is no tool to comprise various Object Oriented Design metrics to determine the quality of Object Oriented Program. The proposed tool provides different features to compute project level metrics, module level metrics and class level metrics for Object Oriented Programming.

II. Literature Survey

Object Oriented Analysis and Design methods, Object Oriented languages, and Object Oriented development environments are currently popular worldwide in both small and large software organizations. The Object Oriented methodology has created new challenges for software companies which use product metrics tool for monitoring, controlling and improving the way they develop and maintain software. The studies have concluded that" traditional" product metrics are not sufficient for characterizing, assessing, and predicting the quality of Object Oriented software systems. To address this issue, OO metrics have recently been proposed in the literature [3], [5], [6]. *Object-Oriented Design Metrics Using UML Class Diagrams* describes a brief survey of object-oriented design considerations. A software design metric approach, known as Factor-Criteria-Measurement, is enhanced to adopt the use of Unified Modeling Language (UML) class diagrams to derive useful object-oriented design metrics. A structural method to present the metrics is also discussed [1]. The Software Assurance Technology Center's (SATC) approach was to select OO metrics that apply to the primary, critical constructs of OO design. The suggested metrics are supported by most literature and also now found in some Object Oriented tools. The metrics evaluate the OO concepts: methods, classes, cohesion, coupling, and inheritance. The metrics focus on internal object structure, external measures of the interactions among entities, measures of the efficiency of an algorithm and the use of machine resources, and the psychological measures that affect a programmer's ability to create, comprehend, modify, and maintain software. *"Object Oriented Design*

Quality Metrics " [11] proposes strategies on how analysis of source code with metrics can be integrated in an ongoing software development project and how metrics can be used as a practical aid in code- and architecture investigations on existing systems. [2] Proposed work focused on defining 12 categories and a set of metrics for each category in order to assess software suitability to meet user's needs. A possible extension of proposed work to measure the forecasted success of open source projects is discussed. A key idea of the proposed extension is to systematically use quantitative metrics that can be automated, taking advantage of the information that can be found in the Web. A neuro-fuzzy approach together with statistical techniques to reveal the

relationship between metrics and dependent variables, and the correlations among those metrics also has to be considered. Customized neuro-fuzzy approach, validate design metrics like CK metrics suite and also investigate other process and personnel metrics [7]. Jie Xu introduced the basic metric suite for Object Oriented Design. The need for such metrics is particularly acute when an organization is adopting a new technology for which established practices have yet to be developed [8]. An improved hierarchical model for the assessment of high-level design quality attributes in object-oriented designs is proposed [4]. In this model, structural and behavioral design properties of classes, objects, and their relationships are evaluated using a suite of object-oriented design metrics. This model relates design properties such as encapsulation, modularity, coupling, and cohesion to high-level quality attributes such as reusability, flexibility, and complexity using empirical and anecdotal information. Empirical evidence supports the role of OO Design complexity metrics, specifically a subset of the Chidamber and Kemerer suite, in determining software defects. Robert Martin proposed a set of metrics that can be used to measure the quality of an object oriented design in terms of the interdependence between the subsystems of the design [14].

III. Outline

This paper is organized as follows. Section 4 presents the proposed work and different metrics namely Project Wide Metrics, Module Wide Metrics, C.K. Metrics Object Oriented Metrics and Quality Attributes. Section 5 and 6 concludes the paper by presenting results, conclusion and future work.

IV. Proposed Work

The proposed system assesses various software design metrics for Object Oriented program like C++ and Java. Metrics are considered as quality indicators of software development process. The metrics assessed in this system are project level metrics, module level metrics and class level metrics. The project level metrics mainly includes reuse ratio, specialization ratio, and average inheritance depth. The module level metrics namely total lines of code, physical lines of code, number of statements, number of blank lines, number of comment lines, Count of all lines Non-comment Non-blank (NCNB) and Executable Statements (EXEC).Class level metrics namely Weighted methods per class (WMC) -Class/Method, Response for a class (RFC) -Class/Message, Lack of cohesion of methods (LCOM) -Class/Cohesion, Coupling between objects (CBO) –Coupling, Depth of inheritance tree (DIT) –Inheritance, Number of children (NOC) –Inheritance. In addition to that the developed system has proposed to compute various design quality attributes. Those design quality attributes are coupling cohesion, message size, abstraction, inheritance, polymorphism, encapsulation, reusability and flexibility.

A. Metrics Description

1) Project Wide Metrics

1.1. Reuse Ratio (U):

The reuse ratio (U) is given by U=number of super class/total number of class.

1.2. Specialization Ratio(S):

This ratio measures the extent to which a super class has captured abstraction. S=number of subclass/number of super class.

1.3. Average Inheritance Depth:

The inheritance structure can be measured in terms of depth of each class with in its hierarchy. Average inheritance depth =sum of depth of each class/number of class.

2) Module Wide Metrics

2.1. Lines of code (LOC): This measure provides a count of total number of lines in the module. It includes source lines, blank lines, comment lines.

2.2. Physical lines of code: This measure provides a count of total number of source lines in the module.

2.3. Number of statements: This measure indicates total number of statements in the module. It includes if, else, switch, case, while, do while, for statements.

2.4. Comment lines: This measure indicates total number of comment lines in a module.

2.5. Blank lines: This measure indicates total number of blank lines in a module.

*2.6. Non-comment Non-blank (NCNB)***:** This measure provides count of all lines that are not comments and not *blanks.*

2.7. Executable Statements (EXEC): This measure provides a count of executable statements regardless of number of physical lines of code.

3) C.K. Metrics Object Oriented Metrics

3.1. Weighted Methods per Class: It is an average of the number of methods within the classes of the software system for which the metrics are being collected. Average Number of Methods per Class reflects the degree of responsibility attributed to a class i.e., it is a predictor of how much time and effort is required to develop and maintain the class. A large number of methods mean a greater potential impact on its derived class, since the derived class inherits all the method of the base class. A class in a system should not be overloaded with large number of functions.

3.2. Depth of Inheritance Tree (DIT): It is a *class* level design metric. The DIT for a particular class calculates the length of the inheritance chain from the root to the level of this class. Deep trees promote reuse because of method inheritance. High DIT increase faults. Most fault-prone classes are the ones in the middle of the tree. A recommended DIT is five or less.

3.3. Number of Children: It indicates the number of immediate subclasses of a class. NOC is counted via the inherited statement. NOC is equal to number of immediate child classes derived from a class via the inherited statement. It measures the breadth of a class hierarchy. High NOC indicates high reuse, fewer faults.

3.4. Coupling between object Classes: It is the number of classes to which a class is coupled. Two classes are coupled when methods declared in one class use methods or instance variables defined by the other class. High CBO is undesirable. High coupling indicates fault-proneness. Only methods and

variable references are counted. Other types of reference, such as use of constants, calls to API declared, handling of events, use of user defined types, and object instantiations are ignored. If a method call is polymorphic (either because of overrides or overload) all the classes to which the call can go are included in the coupled class.

3.5. Response for a Class (RFC): It is a *class* level design metric. Response for a class is a set of methods that can potentially be executed in response to a message received by an object of that class. Response for a class defines the number of methods available in the class. The number of available methods in the class is the sum of the number of local methods in the class and the number of methods called by the local methods. Larger values for this metric imply that more testing is required for that class because of the increased coupling issues.

3.6. Lack of Cohesion (LCOM): It is a class level design metric. It measures the number of disjoint sets of local methods. Disjoint sets are a collection of sets that do not intersect with each other. Any two methods in one disjoint set access at least one common local instance variable.

3.7. Structural Complexity: Complexity is calculated by number of predicated node or decision plus one. Decision include conditional statements like if, else, select, for, do loop, while and end loop.

3.8. Hierarchical Model: Hierarchical Model is used to determine the overall quality of Object Oriented system by evaluating the structural and the behavioral design properties of classes and their relationship. Design properties include encapsulation, polymorphism, modularity, coupling, cohesion, etc.,.

3.9. Abstraction: Abstraction is a measure of the generalization-specialization aspect of the design. Classes in a design which have one or more descendants exhibit this property of abstraction.

3.10. Cohesion: Assesses the relatedness of methods and attributes in a class, strong overlap in the method parameters and attributes types in an indication of strong cohesion.

3.11. Complexity: Complexity is a measure of the degree of difficulty in understanding and comprehending the internal and external structure of classes and their relationships.

3.12. Composition: Measures the "part-of", "has","consist- of" or "part-whole" relationship, which are aggregation relationships in an object –oriented design.

3.13. Coupling: Define the interdependency of an object on other objects in a design. It is a measure of the number of other objects that would have to be accessed by an object in order for that to function correctly.

3.14. Design Size: A measure of the number of classes used in a design.

3.14.1. Encapsulation: Defined as enclosing of data and behavior within a single construct .In Object Oriented design the property specifically refers to designing classes that prevent access to attribute declarations by defining them to be private, thus protecting the internal representation of the objects.

3.14.2. Hierarchies: Hierarchies are used to represent different generalization –specialization concepts in a design. It is the count of the number of non-inherited classes that have children in a design.

3.14.3. Inheritance: A measure of "is –a" relationship between classes. This relationship is related to the level of nesting of classes in an inheritance hierarchy.

3.14.4. Messaging: A count of number of public methods that is available as services to other classes. This is a measure of services that a class provides.

3.14.5. Polymorphism: Polymorphism refers to the ability to substitute objects whose interfaces match for one another at run time. It is a measure of services that are dynamically determined at run time in an object.

3.15. Quality Attributes:

3.15.1. Reusability: Reusability means reflects the presence of OO Design characteristics that allow a design to be reapplied to new problem without significant. Reusability formula= (-0.25*coupling) + (0.25*cohesion) + (0.5*messaging) + (0.5*design size).

3.15.2. Flexibility: Characteristics that allow the incorporation of change in a design. The ability of a design to be adapted to provide Functionality related capabilities. Flexibility formula= (0.25*encapsulation)-(0.25*coupling) + (0.5*composition) + (0.5*polymorphism).

3.15.3. Understandability: The properties of the design that enable it to be easily learned and comprehend. Understandability formula= (-0.33*abstraction) + (0.33*encapsulation)-(0.33*coupling) + (0.33*cohesion)-(0.33*polymorphism)-(0.33*complexity)-(0.33*design size).

3.15.4. Functionality: The responsibilities assigned to the classes of design, which are made available by the classes through their public interfaces. Functionality formula= (0.12*cohesion) + (0.22*polymorphism) + (0.22*messaging) + (0.22*design size) + (0.22*hierarchies)

3.15.5. Extendibility: It refers to the presence and usage of properties in an existing design that allow for the incorporation of new requirements in the design. Extendibility formula = (0.5*Abstraction)-(0.5*coupling) + (0.5*inheritance) + (0.5*polymorphism).

3.15.6. Effectiveness: It refers to a design's ability to achieve the desired functionality and behavior using OO Design concepts. Effectiveness formula= (0.2*abstraction) + (0.2*encapsulation) + (0.2*composition) + (0.2*inheritance) + (0.2*polymorphism)

V. RESULTS

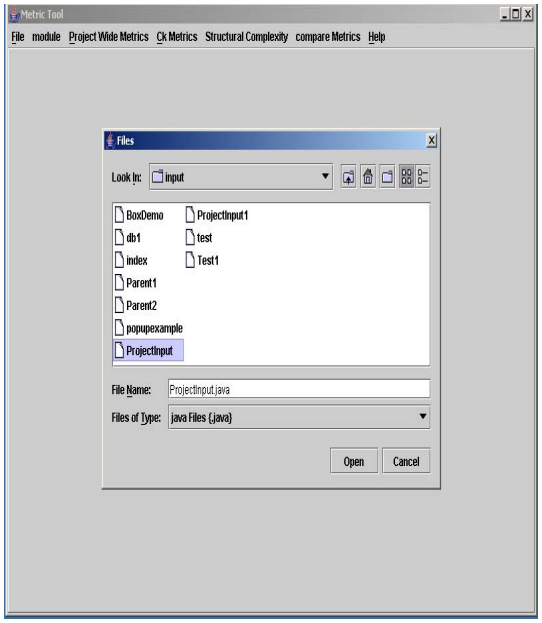

Fig .2 Screenshot Shows the Choosing the Input File From the File Menu

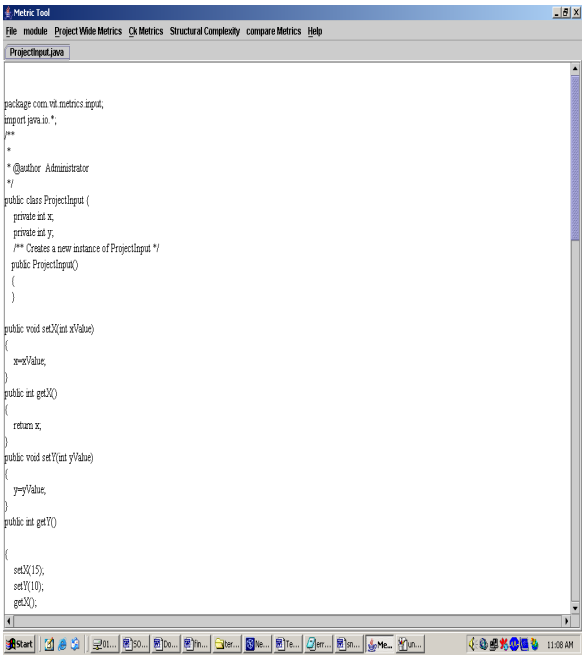

Fig.3. Screenshot Shows Content Display for the Given Program

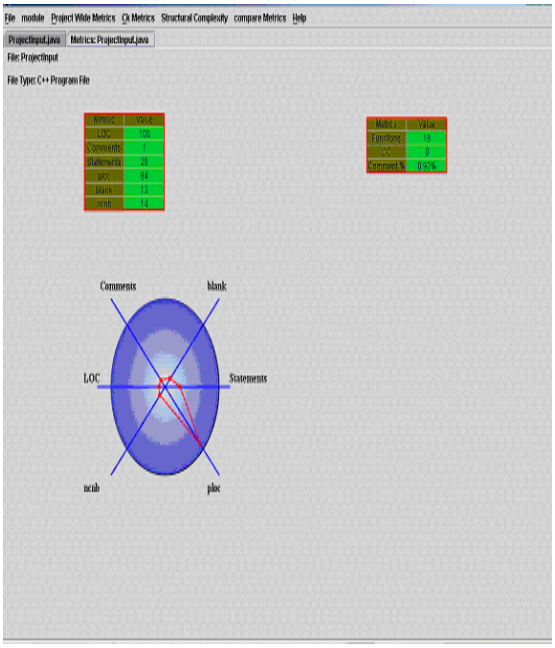

Fig.4 Screenshot shows the Module Wide Metrics details for the given input program.

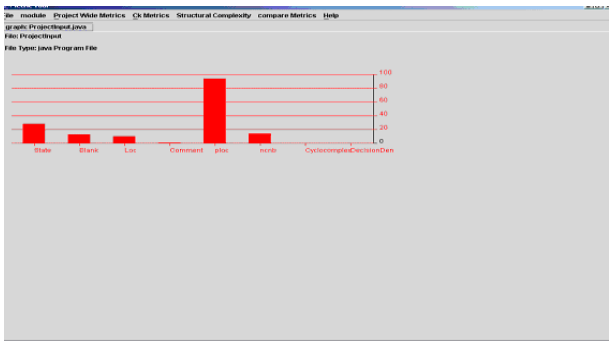

Fig.5 Screenshot shows the Module Wide Metrics details for the given input program.

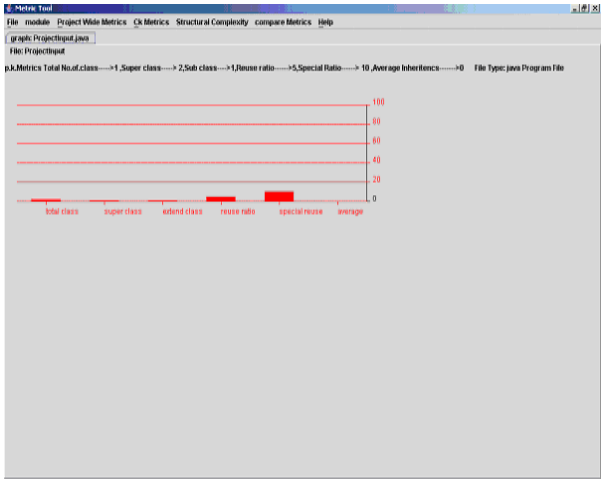

Fig.5. Screenshot shows the Project Wide Metrics details for the given input program.

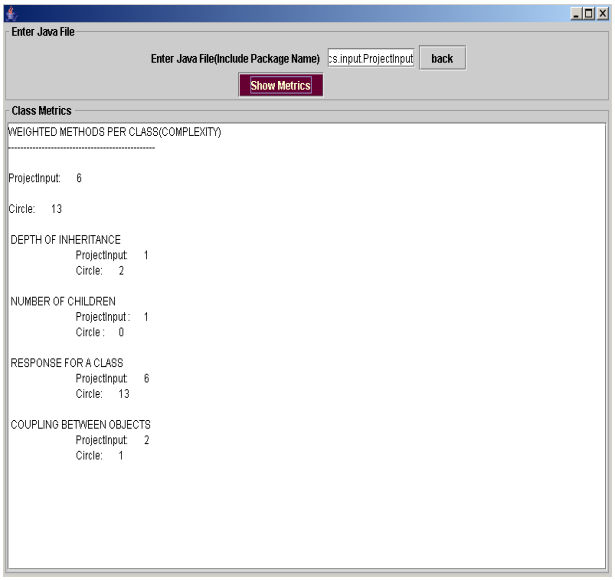

Fig.6. Screenshot shows the C.K .Metrics details for the given input program.

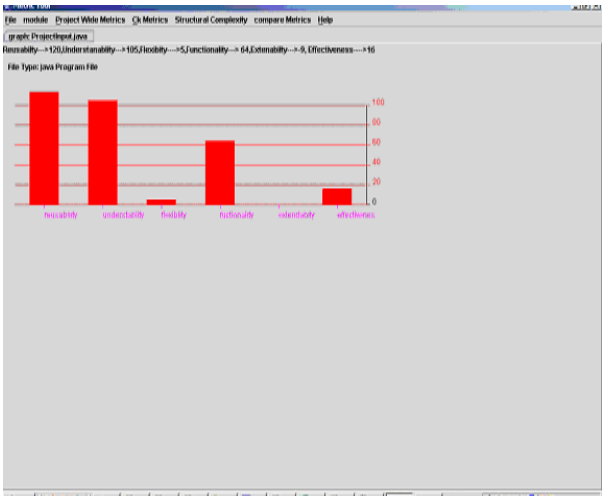

Fig.7. Screenshot shows the Hierarchical metrics details for the given input program

VI. Conclusion And Future Work

The developed system is a comprehensive tool which calculates project wide metrics, module wide metrics, chidember kemerer metrics, design quality attributes and hierarchical metrics for Object Oriented program. The Assessment tool is used to assess the quality of the java application. The tool can be enhanced to assess other Object Oriented languages such as C++, C sharp, etc., The enhanced assessment tool will be helpful in assessing the quality in advance to develop awareness for quality issues such as reliability, testability and maintainability.

References

[1] Albert Dieter Ritzhaupt , "Object-Oriented Design Metrics Using UML Class Diagrams " , 2nd CCEC Symposium 2004, April 8, 2004, UNF, Jacksonville, FL. USA.

[2] Alessandra Cau, Giulio Concas, Michele Marchesi "Extending Open BRR with Automated Metrics to Measure Object Oriented Open Source Project Success", May 19, 2006.

[3] F.B. Abreu and R. Carapuca, "Candidate Metrics for Object-Oriented Software within a Taxonomy FraFramework," . System and Software, vol. 26, no. l, pp. 87-96, Jan. 1994

[4] J.Bansiya and C.G.Davis, "A Hierarchical Model for Validation of Object Oriented Design Quality Assessment", IEEE Transactions on Software Engineering, Vol.28, No 1, January, 2002.

[5] J M Bieman and B.-K. Kang, "Cohesion and Reuse in an Object-Oriented System," Proc ACM SIGSOFT Symp Software reusability, Seattle, Wash., pp. 259-262,1995.

[6] S.R.Chidamber and C.F.Kemerer, "A Metrics Suite for Object Oriented Design", IEEE Transactions on Software Engineering, Vol.20, No 1, Jan, 1994.

[7] Fernando Brito e Abreu "Design Metrics for Object –Oriented Software Systems".

[8] Jie Xu, Danny Ho and Luiz Fernando Capretz "An Empirical Validation of Object Oriented Design Metrics for Fault Prediction", Journal of Computer Science 4 (7): 571-577, 2008, ISSN 1549-3636, © 2008 Science Publications.

[9] Jaana Lindroos , "Code and Design Metrics for Object-Oriented Systems",

[10] Jaaksi A., "A Method for Your First Object-Oriented Project", JOOP, January 1998.

[11] Magnus Andersson Patrik Vestergren,"Object Oriented Design Quality Metrics "Uppsala Master's Theses in Computer Science 276 ,2004-06-07 ,ISSN 1100-1836.[1100].

[12] Ms Puneet Jai kaur, Ms Amandeep Verma , Mr. Simrandeep Thapar3, "Software Quality Metrics for Object-Oriented Environments ",Proceedings of National Conference on Challenges & Opportunities in Information Technology (COIT-2007) ,RIMT-IET, Mandi Gobindgarh. March 23, 2007.

[13] Ramanath Subramanyam and M.S. Krishnan "Empirical Analysis of CK Metrics for Object-Oriented Design Complexity Implications for Software Defects ", IEEE Transactions on Software Engineering, Vol. 29, No. 4, April 2003.

[14] Robert Martin," OO Design Quality Metrics an Analysis of Dependencies" October 28, 1994.

[15] Victor R .Basili, Fellow, "A Validation Of object Oriented Design Metrics as Quality Indicators", IEEE Transactions on Software Engineering, Vol.22, No 10, October, 1996.

Prof. Mythili Thirugnanam is currently working as Assistant Professor, School of Computing Science and Engineering, Vellore Institute of Technology, Vellore, India. She is having 2 years of research and 3 years of teaching experience. She has pursued her M.S degree in Software Engineering from VIT University, Vellore. She is currently pursuing Ph.D in VIT University. She is member of CSI. Her research interest includes Software Quality and Digital Image Processing.

Prof. Swathi.J.N. is currently working as Assistant Professor ,School of Computing Science and Engineering , Vellore Institute of Technology, Vellore, India. She is having 4 years of teaching experience. She has pursued her M.S. degree in Software Engineering from VIT University, Vellore. She is currently pursuing Ph.D in VIT University. Her research interest includes Software Quality and Data Mining.

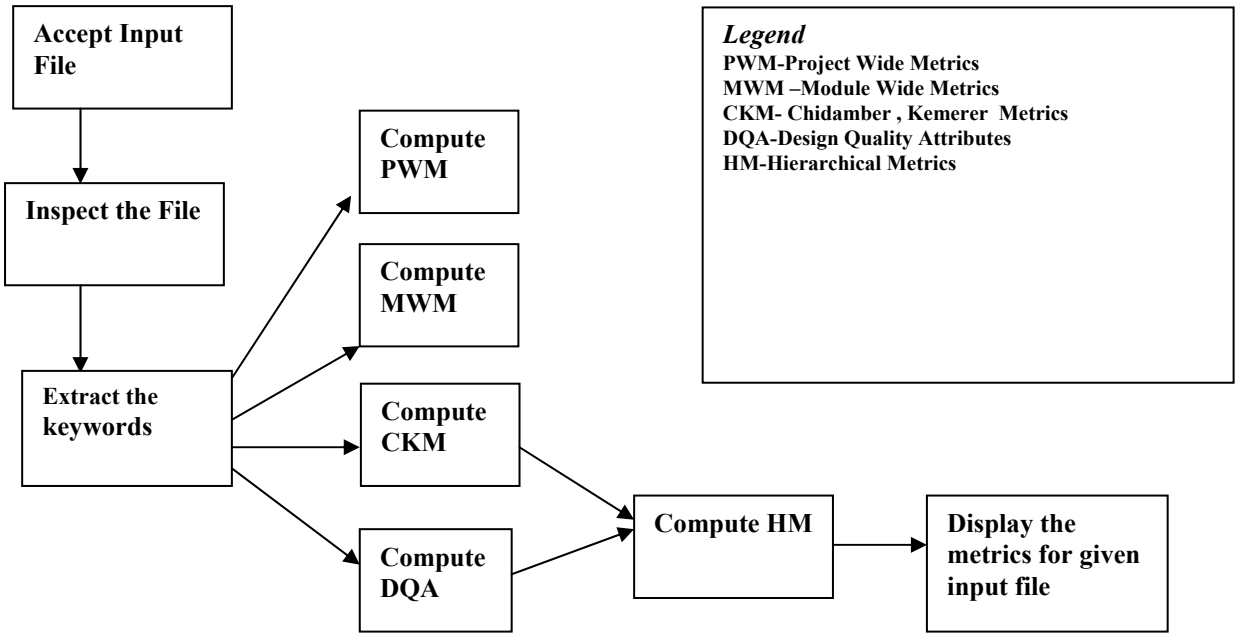

Fig .1 Overview of the Proposed System