Estimation of Software Size by Using UML- A Review

1Shefali Chourasia, 2Sapna Choudhry Jain
1Master Student, 2Professor and Head,
Department of Computer Science Engineering
SRGI, Jabalpur

Abstract: **As we know that in Software Engineering, measuring the software is an important activity. For measuring the software appropriate metrics are needed. Using software metrics we are able to attain the various qualitative and quantitative aspects of software. Software Metrics are a unit of measurement to measure the software in terms of quality, size, efforts, efficiency, reliability, performance etc. Measures of specific attributes of the process, project and product are used to compute software metrics.**

INTRODUCTION

The objectives of this research are to make an empirical evaluation of software size metrics based on UML with the help of two case studies and then calculate that empirical data consisting of actual values and thereby showing that how the software size metrics will be derived from an UML model via Class Diagrams and the below listed interaction diagrams.

1. Activity Diagrams
2. State chart Diagrams
3. Component Diagrams
4. Collaboration Diagrams

For carrying out this research, two real case studies namely (i) Virtual Class Room and (ii) Data Secrecy System will be taken for practical evaluation. The UML modeling of these systems will be done and the software size metrics of these systems will be evaluated based on the UML models, using the non-functional techniques (LOC, FP, and COCOMO-II).The metrics will be specified using UML extension mechanism and then will be calculated with the help of a tool. The estimated values will be compared with the actual software. Thus, the aim of our research is to evaluate the empirical value sets of UML models and thereby, showing the use of various size metrics and validate their extraction procedure from UML design with the help of interaction diagrams.

Brief History and Development of UML:

The UML effort started officially in October 1994 when Ram Baugh joined Booch at Rational. The version 0.8 draft of the Unified Method (as it was then called) was released in October 1995.Around the same time, Jacobson joined rational and the scope of the UML project was expanded to incorporate OOSE and then was the release of version 0.9 documents in June 1996.With the establishment of an UML consortium, with several organizations ,came into light was the UML version 1.0. And UML 1.0 was offered for standardization to the OMG in January 1997.A revised version of UML1.1 was offered to the OMG for standardization in July 1997 and it was finally accepted by OMG in September 1997.and thereby, UML 1.1 was adopted by the OMG in November 14, 1997. Hence forth, UML was maintained by OMG Revision Task Force, which produces versions 1.3, 1.4 and 1.5.From 2000 to 2003, an updated specification UML 2.0 was produced .After, being reviewed by Finalization Task Force (FTF) for a year, the official version of UML 2.0 was adopted by OMG in early 2005.The actual UML specification documents are found on the OMG website at www.omg.org [4].

Introducing UML 2.0 Diagrams:

According to the latest UML 2.0 the diagrams are divided into three categories [5]:

1. **Structural Diagrams:** These include the Class Diagram, Object Diagram, Component Diagram, Composite Structure Diagram, Package Diagram, and Deployment Diagram.
2. **Behavior Diagrams:** include the Use Case Diagram (used by some methodologies during requirements gathering), Activity Diagram, and State Machine Diagram.
3. **Interaction Diagrams:** all derived from the more general Behavior Diagram, include the Sequence Diagram, Communication Diagram, Timing Diagram, and Interaction Overview Diagram.

Software Sizing Metrics for Object-oriented systems:

The metrics that support the structural programming concepts can also be applied to the Object- oriented concepts. The Object-oriented Methodology focuses on objects. Object-orientation is a technique for system modeling. Object-orientation is ideally suited to creating models of real systems, and especially for simulating the system [6].

Related Work

Many scientists and researchers have studied the software metrics based on UML models. And thus have given their immense contributions to the field of research in the computer sciences .A lot of work has been done till date in the area of research while considering software metrics related to UML design.

In their paper **Tong Yi et al. [7]** analyzed and compared some typical metrics for UML class diagrams from different viewpoints , different types of relationships, different types of metric values, complexity, and perfume theoretical and empirical validation. They have tried to analyze the existing popular metrics for UML class diagrams both theoretically and experimentally from several different viewpoints. The analysis

shows that most current metrics have their shortcomings while being effective or efficient for some special characteristics of the system.

Li Wei et al. [8] have presented an empirical study of OO metrics in two iterative processes: the short-cycled agile process and the long-cycled framework evolution process. They have found that OO metrics are effective in predicting design efforts and source lines of code added, changed, and deleted in the short-cycled agile process and ineffective in predicting the same aspects in the long-cycled framework process. This leads them to believe that OO metrics' predictive capability is limited to the design and implementation changes during the development iterations, not the long-term evolution of an established system in different releases.

Mitchell et al. [9] presented a position paper outlining a programmed of research based on the quantification of run-time elements of Java programs. In particular, we adapt two common object-oriented metrics, coupling and cohesion, so that they can be applied at run-time. The results presented in this paper are of a preliminary nature, and do not provide a justifiable basis for generalization. However, she believed that they do provide an indication that the evaluation of software metrics at run-time can provide an interesting quantitative analysis of a program.

Through their paper **Christodoulakis et al. [10]** have derived the results on metrics used in object oriented environments. Their survey includes a small set of the most well known and commonly applied traditional software metrics which could be applied to object–oriented programming and a set of object–oriented metrics (i.e. those designed specifically for object–oriented programming). These metrics were evaluated using existing meta–metrics as well as meta–metrics derived from our studies, based mostly on the practitioner's point of view, and emphasizing applicability in three different programming environments: Object Pascal, C++ and Java.

In this paper **M.Das et al. [11]** have stated that Component-Based Software Engineering (CBSE) has shown significant prospects in rapid production of large software systems with enhanced quality, and emphasis on decomposition of the engineered systems into functional or logical components with well-defined interfaces used for communication across the components. In this paper, a series of metrics proposed by various researchers have been analyzed, evaluated and benchmarked using several large-scale publicly. Available software systems. A systematic analysis of the values for various metrics has been carried out and several key inferences have been drawn from them. A number of useful conclusions have been drawn from various metrics evaluations, which include inferences on complexity, reusability, testability, modularity and stability of the underlying components. The inferences are argued to be beneficial for CBSE-based software development, integration and maintenance.

Jamali [12] has stated the central role that software development plays in the delivery and application of information technology, managers are increasingly focusing on process improvement in the software development area. The focus on process improvement has increased the demand for software measures, or metrics with which to manage the process. 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. He has addressed these needs through the development and implementation of a suite of metrics for OO design.

Shaik Amjan, et al. [13] has presented the obtainable and new Software metrics useful in the different phase of the Object-Oriented Software Development Life Cycle. Metrics are used by the software industry to itemize the development, operation and maintenance of software .They have presented metrics for Object-oriented Software systems. A mechanism is provided for comparing measures , which examine the same concepts in different ways , and facilitating more rigorous decision-making, regarding the explanation of new measures and the selection of existing measures for a specific goal of measurement.

Jahan Vafaei et al. [14] has proposed a new method to calculate the complexity of UML models focusing on UC and UUCP. They have used regression based model to build size estimation from complexity design measures and is concluded that some of the metrics such as attribute, method , inheritance and communication between classes with regard to use communication between classes with regard to use case scenario of each class has fairly accurate prediction for size estimation.

Chen Yue Boehm et al. [15] has also stated size as an important metric for UML models via SLOC and FP. In this paper, 14- project studied of three different sizing metrics which cover different software lifecycle activities. Their results show that the software size in terms of SLOC was moderately well correlated with the number of external use cases and number of classes.

Linda Edith .P et al. [16] has integrated the different Object-oriented metric tools and made them available as a single add-on. The main aim is to search some important tools and making them as single add-on to provide the user-friendly environment. In this paper are presented, a brief discussion of a number of algorithms with a comparative study of a few significant ones based on their performance and memory usage.

In their paper, **Subramanyam Ramanath et al. [17]** has laid an emphasis on the design aspects of high quality Object-oriented(OO) applications. They have also provided empirical evidence supporting the aspects of OO design complexity metrics .They have also found that the effects of these metrics on defects are different across the samples from two different programming languages-C++,Java.

Tegarden P. David et al. [18] in their paper has given an acceptable measure of software quality and quantify software complexity. Also, their research reports the effects of polymorphism and inheritance on the complexity of object-oriented systems as measured by the traditional metrics. Their research indicates the results that the traditional metrics are applicable to the measurement of the complexity of object-oriented systems.

Through their paper, **Doban Orysolya et al. [19]** stated the experiments of a pseudo code estimation are detailed with a specific emphasis on the cost estimation driven system design .They have also stated that in the field of dependability this methodology can be used to analyze solution alternatives and may play a central role in the current trend of constricting dependable systems from COTS elements by focusing effort to the crucial parts of the target system.

Lavazza Luigi et al. [20] in her paper has reported that the estimation of the development effort of real-time software using Function – Points as the means for expressing the size of the software. The immediate goals were the identification of a reliable sizing method for estimation-oriented software sizing, and the adoption of an estimation methodology and tool.UML – based counting of function points appears as effective and efficient , providing a systematic and unambiguous approach to measurement.

In their paper **Chidamber et al. [21]** has made an analysis of a set of metrics proposed by Chidamber and kemerer(1994) is performed in order to assess their usefulness for practicing managers. Also , the empirical results suggest that the metrics provide significant explanatory power for variations in these economic variables, over and above that provided by traditional measures, such as size in lines of code, and after controlling for the effects of individual developers.

Russo Barbara et al. [22] has stated in their paper a very simple measure, the number of method s of an analysis object is well and significantly co-relate with the size of final products in both the data sets taken .In her paper, she has used Class Metrics : external complexity and internal complexity, depth of inheritance tree, number of children and number of attributes.

Through their paper, **Doban Orsolya et al. [23]** has stated that the main objective of software project management is to assure that a software product will be delivered in time, keeping the cost limits and a proper quality. Thus, their main objective was to provide a methodology , which is able to take into generate the mathematical model of a software process optimization problem based on UML diagrams.

Kumar Rakesh et al. [24] have highlighted in their paper that the object-oriented software metrics proposed in 90's by Chidamber, kermerer and several studies were conducted to validate the metrics and discovered several deficiencies. In their paper, they have made a comparative study between the metrics proposed by Chidamber, kermerer and Li.

G. Costagliola et al. [25] has stated in their paper FP-like approach , named class point to estimate the size of object-oriented products. In particular, two measures are proposed , which are theoretically validated showing that they satisfy well-known properties necessary for size measures. An initial , empirical validation is also performed , meant to assess the usefulness and effectiveness of the proposed measure to predict the development effort of Object-oriented systems. Moreover, a comparative analysis is carried out, taking into account several other size measures.

Thapilyal et al. [26] in their paper has stated that the Object Oriented design, today, is becoming more popular in software development environment. Object oriented measurements are being used to evaluate and predict the quality of software. But an Object Oriented metric is able to treat function and data as combined integrated object. In this paper they have evaluated two metrics Weighted Method per Class (WMC) and Coupling between Object Classes (CBO) of Chidamber and Kemerer metrics Suite. They have done an empirical study and tried to find out the nature of relationship of these metrics with defects. In other words, it has been investigated whether these metrics are significantly associated with defects or not. They have deliberately taken different projects & tried to check if these metrics can really be reliable measurements for predicting defects when applied to inherently different projects.

In this paper, **Kim Hyoseob et al. [27]** has proposed some new software metrics that can be applied to UML modeling elements like *classes* and *messages*. These metrics can be used to predict various characteristics at the earlier stages of the software life cycle. A CASE tool is developed on top of *Rational Rose*1 using its Basic Script language and we provide some examples using it. In this paper, some new software metrics were introduced that can be used at the early stages of the software life cycle. They measure various characteristics of model, class, message, and use case in the future, these metrics need to be evaluated using data from the industry.

In their paper **Jurjens Jans et al. [28]** has stated that dependable systems have to be developed carefully to prevent loss of life and resources due to system failures. This paper gives an overview of reliability-related analyses for the design of component-based software systems. This enables the identification of failure-prone components using complexity metrics and the operational profile, and the checking of reliability requirements using stereotypes. They report on the implementation of checks in a tool inside a framework for tool-supported development of reliable systems with UML and two case studies to validate the metrics and checks.

Gandhi Parul et al. [29] stated that Object-oriented metrics plays an import role in ensuring the desired quality and have widely been applied to practical software projects. The benefits of object-oriented software development increasing leading to development of new measurement techniques. Assessing the reusability is more and more of a necessity. Reusability is the

key element to reduce the cost and improve the quality of the software. Generic programming helps us to achieve the concept of reusability through C++ Templates which helps in developing reusable software modules and also identify effectiveness of this reuse strategy. The advantage of defining metrics for templates is the possibility to measure the reusability of software component and to identify the most effective reuse strategy. In this paper they have proposed four new independent metrics Number of Template Children (NTC), Depth of Template Tree (DTT) Method Template Inheritance Factor (MTIF) and Attribute Template Inheritance Factor (ATIF), to measure the reusability for object-oriented systems.

Rosenberg Linda et al. [30] in their paper has discussed that how the NASA projects, in conjunction with the SATC, are applying software metrics to improve the quality and reliability of software products. Reliability is a by-product of quality, and software quality can be measured. They will demonstrate how these quality metrics assist in the evaluation of software reliability. They concluded with a brief discussion of the metrics being applied by the SATC to evaluate the reliability .

Berardinelli Luca et al. [31]. in their paper has stated that the modeling and validation of Non-Functional Properties (NFPs) is a crucial task for software systems to satisfy user expectations then for software projects to succeed. Nevertheless this research field still suffers the heterogeneity of hermetic approaches aiming to the modeling and validation of one single non-functional property without sharing information among them and loosing the view of the system as a whole. In this paper they presented preliminary results on modeling and analysis of different NFPs starting from a single UML model, suitably extended with profiles like MARTE and DAM. To support the validity of modeling they have shown how the approach allows the derivation of Petri Net, Queuing Network and Fault Tree models for analyzing, respectively, availability, performance and reliability indices of a software system under development.

Thirugnanam Mythili et al. [32] in their paper has stated that metrics measure certain properties of a software system by mapping them to numbers (or to other symbols)according to well-defined, objective measurement rules. 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.

Monperrus Martin et al . [33] has given that metrics offer a practical approach to evaluate properties of domain-specific models. However, it is costly to develop and maintain measurement software for each domain specific modeling language. In this paper, we present a model-driven and generative approach to measuring models. The approach is completely domain-independent and operational zed through a prototype that synthesizes a measurement infrastructure for a domain specific modeling language. This model-driven measurement approach is model-driven from two viewpoints: 1) it measures models of a domain specific modeling language; 2) it uses models as unique and consistent metric specifications, wart. a metric specification meta model which captures all the necessary concepts for model-driven specifications of metrics. The benefit from applying the approach is evaluated by four case studies. They indicate that this approach significantly eases the measurement activities of model-driven development processes.

Shatnawi Raed [34] has stated in his paper that there is a dearth of studies that identified thresholds values of Chidamber and Kemerer (CK) metrics. In an empirical study on open-source software, Eclipse project—Version 3.0, we identified the thresholds values for CBO, RFC and WMC at two levels of risks using a quantitative methodology based on the logistic regression curve. These threshold values can be used to identify the most error-prone classes.

Zivkovic Ales et al. [35] has stated in their paper a unified mapping of UML models into function points. The mapping is formally described to enable the automation of the counting procedure. Three estimation levels are defined that correspond to the different abstraction levels of the software system. Through this paper, the gap between an object abstraction and the FPA abstraction was fulfilled via unified mapping based on four existing mappings. One important contribution was a modified complexity table based on OO metrics that defined less data elements to achieve the same complexity.

Arisholm [36] has given that a common way to quantify the coupling is through static code analysis. However, the resulting static coupling measures only capture certain underlying dimensions of coupling. Other dependencies regarding the dynamic behavior of software can only be inferred from run-time information. This paper describes how several dimensions of dynamic coupling can be calculated by tracing the flow of messages between objects at run-time. Preliminary results suggest that dynamic coupling may also be useful for developing prediction models and tools supporting change impact analysis.

In their paper , **Lange Christian et al. [37]** has stated that the use of metrics as means of control and improvement plays an important role in software engineering. They have proposed a definition of completeness of a UML model and presented a set of rules to assess model completeness. They have also reported results from industrial case studies to assess the level of completeness in practice. The amount of completeness and consistency rule violations was very high. This paper has defined a collection of rules that capture completeness

constraints in terms of a meta-model that covers multiple views and makes it clear that the use of tools for identifying completeness violations can aid in enforcing more uniform design conventions.

Through their paper , **Uemura Takuya et al. [38]** has presented the detailed function-point analysis measurement rules using design specifications based on the Unified Modeling Language and described a function-point measurement tool, whose inputs are design specifications developed on Rational Rose_. Also, they have reported the tool validation work on software involved in software evolution at an organization where they have applied the tool to actual design specifications and examined the differences between the function point values obtained by the tool and those of an experienced function point measurement specialist at the organization.

Through their paper, **Dagpinar Melis et al. [39]** have designed and conducted an empirical study based on historical data collected from the maintenance history of a medium-sized object-oriented system. Indirect coupling has also been taken into account in their work in order to evaluate its impact. They have used multivariate regression analysis to select a suite of metrics that serves as best predictors for maintainability. Their study provides empirical evidence that object-oriented metrics can effectively be used to predict maintainability of software systems.

In their paper, **Genero Marcela et al. [40]** has presented a set of metrics -based on UML relationships- which measure UML class diagram structural complexity following the idea that it is related to the maintainability of such diagrams. Also summarized are two controlled experiments carried out in order to gather empirical evidence in this sense .Quantitative measurement instruments are useful to assess class diagram quality in an objective way, thus avoiding bias in the quality evaluation process. The results obtained in both experiment shows that most of the metrics they have proposed (NAssoc, NAgg, NaggH, MaxHAgg, NGen, NgenH and MaxDIT) are good indicators of class diagram maintainability sub-characteristics. Performing empirical validation with the metrics is fundamental in order to demonstrate their practical utility.

Genero Marcela et al. [41] in their paper have introduced and analyze a set of an existent object oriented metrics that can be applied for assessing class diagrams complexity at the initial phases of the object oriented development life cycle. .They have presented a state of the art in OO metrics that can measure the complexity of UML class diagrams obtained in the initial phases of the OO development life cycle. Analyzing several proposals, they deduce that there is a gap in OO metrics related to relationships, such as association, aggregation and dependency. They, therefore propose new metrics, to cover this necessity. Their metrics are defined, at different levels of granularity: class and package.

Basili R. Victor et al. [42] in their paper, their goal is to assess Object-oriented Design metrics as predictors of fault-prone classes and, therefore, determine whether they can be used as early quality indicators. To perform their validation accurately, they collected data on the development of eight medium-sized information management systems based on identical requirements. All eight projects were developed using a sequential life cycle model, a well-known 00 analysis/design method and the C++ programming language. Based on empirical and quantitative analysis, the advantages and drawbacks of these 00 metrics are discussed here . They collected data about faults found in object -oriented classes. Based on these data, they verified how much fault-proneness is influenced by internal (e.g., size, cohesion) and external (e.g., coupling) design characteristics of 00 classes.

Through their paper, **Rosenberg H . Linda et al. [43]** has stated that the evaluation of the utility of a metric as a quantitative measure of software quality was based on the measurement of a software quality attribute. It is also stated that the Product Quality for code and design has five attributes. They are Efficiency, Complexity, Understandability, Reusability, and Testability/Maintainability.

Through his paper, **Reißing Ralf [44]** has presented a formal model for object-oriented design called ODEM (Object-oriented Design Model) . This model can serve as a foundation for the formal definition of object-oriented design metrics. ODEM is based on the UML meta-model, that provides a formal model of object-oriented designs expressed in UML, the most widespread design notation. Also, two case studies on existing metrics suites for object-oriented design show the benefits of applying ODEM to established object-oriented design metrics. It also gives the examples of the use of ODEM for defining object-oriented metrics.

Wolff .J et al. [45] in their paper have stated the development and description of an evolutionary algorithm that layouts UML class diagrams. It evolves the layout by mutating the positions of class symbols, inheritance relations, and associations. The process is controlled by a fitness function that is computed from several well-known and some new layout metrics. In their algorithm the control of the evolution does not refer to the fact that we draw UML class diagrams. Elements are randomly chosen for mutation. In the continuation of this work ,they will emphasize the semantics of UML class diagrams, try out more mutations and also consider crossovers.

Derezinska Anna [46] in his paper, has stated that the dependency area of an element of a UML design is a part of the design that is highly influenced by the given initial element. Dependency areas are identified using sets of propagation rules and strategies. Also, this paper is devoted to the specification of the rules and strategies. They are specified using an extended UML meta-model and expressions in the Object Constraint Language (OCL). He has also presented the specification of dependency areas identified in the UML designs. The approach is based on the strategies and propagation rules. The concept of dependency areas provides a basis for controlling changes

within a design, as well as an impact of requirements on the design and the generated code. It can support automation of project management and maintenance.

Through their paper ,**Xu Dianxiang et al. [47]** have stated a UML-based approach to testing whether or not an aspect-oriented program conforms to its expected crosscutting behavior. They have also explored the aspect-oriented UML design models to derive tests for exercising interactions between aspects and classes. Each aspect-oriented model consists of class diagrams, aspect diagrams, and sequence diagrams. For a method under test, they weave the sequence diagrams of the advice on the method into the method's sequence diagram. Based on the woven sequence diagram and class/aspect diagrams, they then generate an AOF (Aspect-Object Flow) tree by applying coverage criteria such as condition coverage, polymorphic coverage, and loop coverage to woven sequence diagrams.The approach can help testers reveal several types of faults that are specific to spectral structures, such as incorrect advice type, strong or weak point cut expressions, and incorrect aspect precedence.

Fernandes M. Joao et al. [48], in their paper have stated that how the functional and the object-oriented views can be inter-played to represent the various modeling perspectives of embedded systems. They have also presented that how the main modeling tool of the traditional structured methods, data flow diagrams, can be integrated in an object-oriented development strategy based on the unified modeling language. The combination of the functional and object-oriented approaches, represented by DFDs and UML, respectively was analyzed. The rationale was to always have as the major model of the implementation phase, some object or class diagram so that an object-oriented programming languages could be used, but also to include DFDs in the modeling process.

Wieringa Roel [49], in his paper has given a methodological framework for software specification based on systems engineering and show how the UML fits into this framework. Also, an essential modeling approach to formalizing the UML within this framework is argued. Finally, transition system semantics for the UML is discussed, that fits this semantics approach.

Webby Richard et al . [50], in their paper have introduced a logical schema for the integration of software process modeling and software measurement. The schema promotes a common understanding of concepts and terminology, serving as a bridge across the fields of process modeling and software metrics. Also a formal unified view of the major information entities and their inter-relationships has been presented. The schema is designed to be general enough to be used with a variety of different process modeling formalisms and metrics approaches. The paper also incorporates several abstraction mechanisms to support the handling of complex process information and multiple evolving versions of that information. The schema should be useful for both researchers and practitioners in empirical process modeling studies of large-scale systems development and evolution.

REFERENCES

[1]. Pressman S. Roger "Software Engineering" Sixth Edition, McGraw Hill International 2005, pg649, chap 22, ISBN : 007-124083-7.

[2]. Rumbaugh James, Jacobson, Ivar and Booch Grady, "The Unified Modeling Language User Guide" Second Edition 2008, pg 5, chap1.

[3]. Rumbaugh James, Jacobson Ivar and Booch Grady" The Unified Modeling Language User Guide" Second Edition2008, pg 6, chap1.

[4]. Rumbaugh James, Jacobson Ivar and Booch Grady" The Unified Modeling Language User Guide" Second Edition 2008. http://www.uml.org

[5]. Jacobsn Magnus Christerson , Patrick Jonsson ,Gunnar Overgaard" Object-oriented Software Engineering" 2008, pg 66, chap3.

[6]. Yi Tong et. al, "A Comparison of Metrics for UML Class Diagrams" ACM SIGSOFT Software Engineering Notes Page 1, September 2004, Volume 29 .

[7]. Li Wei et .al, "An Empirical Validation of Object-Oriented Metrics in Two Different Iterative Software Processes" IEEE Transactions On Software Engineering , November 2003 , Volume 29 NO. 11, 1043.

[8]. Mitchell Aine et. al , "Toward a definition of run-time object-oriented metrics" 7TH ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering , 2003.

[9]. Xenos M. et al. "Object-oriented metrics – a survey" Proceedings of the FESMA 2000, Federation of European Software Measurement Associations, Madrid, Spain, 2000.

[10]. Arasimhan Lakshmi.V et.al, "Evaluation of a Suite of Metrics for Component Based Software Engineering (CBSE)" Issues in Informing Science and Information Technology Volume 6, 2009.

[11]. Amali Mohsen Seyyed, "Object Oriented Metrics", Proceedings of Informatics and Systems, 7th International Conference on march 28-30,2010.

[12]. Shaik Amjan et.al, "Metrics for Object Oriented Design Software Systems: A Survey" Journal of Emerging Trends in Engineering and Applied Sciences (JETEAS) 1 (2): 190-198 c, 2010.

[13]. Jahan Vafaei et.al , "A New Method Software Size Estimation based on UML Metrics", Proceedings of Informatics and Systems, 7th International Conference on march 28-30,2010

[14]. Chen Yue , Boehm Barry et.al , "An Empirical Study of eServices Product UML Sizing Metrics, Proceedings of Informatics and Systems, 7th International Conference on march 36-46, 2010

[15]. Linda Edith P et. al , "Metrics for Component based Measurement Tools", International Journal of Science & Engineering ,Research Volume 2,Issue 5,May -2011

[16]. Subramanyam Ramanath et al, "Empirical Analysis of CK Metrics for Object-oriented Design Complexity:

Implications for Software Defects", IEEE Transactions on Software Engineering, Vol. 29/4, April 2003.

[17]. Tegarden P. David et al., "Effectiveness of Traditional Software Metrics for Object-Oriented Systems", Proceedings of Informatics and Systems, 7th International Conference on march 52-60, 2010.

[18]. Doban Orysolya et. al, "Cost Estimation Driven Software Development Process", International Metals Review,1979, v. 24, pp. 149-173.

[19]. Lavazza Luigi et al., "Using Function Point in the Estimation of Real-Time Software: an Experience", Proceedings 5th Software Measurement European Forum, Milan 2008.

[20]. Chidamber et al., "Managerial use of metrics for Object-oriented software: an exploratory analysis", IEEE Transactions on Software Engineering,Vol.24 ,Issue 8

[21]. Russo Barbara et.al, "Early estimation of software size in Object-oriented environments": a case study in a CMM level 3 software firm.

[22]. Doban Orsolya et al., "UML Based Software Process Management", Periodica Polytechnica Ser.El. Eng Vol.47,No.3-4,PP.213-228(2003)

[23]. Kumar Rakesh et al., "Comparing Complexity in Accordance with Object-oriented Metrics", International Journal of Computer Applications, Article-8,2011.

