UNIQUE FUNDAMENTALS OF SOFTWARE MEASUREMENT AND SOFTWARE METRICS IN SOFTWARE ENGINEERING

Dr. K.P. Srinivasan

Associate Professor in Computer Science, C.B.M. College,
Kovaipudur, Coimbatore – 641 042, Tamil Nadu, India

ABSTRACT

The most important aim of software engineering is to improve software productivity and quality of software product and further reduce the cost of software and time using engineering and management techniques. Broadly speaking, software engineering initiative has been introduced during software crisis period to describe the collection of techniques that apply engineering and management skills to the construction and support of software process and products. There is no universally agreed theory for software measurement. And the software metrics are useful for obtaining the information on evaluation of process and product in software engineering. It helps to plan and carry out improvement in software organizations and to provide objective information about project performance, process capability and product quality. The process capability is extremely important for software industry because the quality of products is largely determined by the quality of the processes. The make use of of existing metrics and development of innovative software metrics will be important factors in future software engineering process and product development. In future, research work will be based on using software metrics in software development for the development of the time schedule, cost estimates and software quality and can be improved through software metrics. The permanent application of measurement based methodologies is used to the software process and its products to provide important and timely management information, together with the use of those techniques to improve that software process and its products. This research paper mainly concentrates on the overview of unique basics of software measurement and exclusive fundamentals of software metrics in software engineering.

KEYWORDS

Software Quality, RBSM, PKM, PEPE, Software Industry, Software Measurement, SEM, Software Metrics, Object Oriented Metrics, Software Development, Software Engineering, Computer Science.

1. INTRODUCTION

The software measurement is an important research subject in computer science [21-28]. According to eminent researchers Jacobson, I., and Seidewitz, E. (2014), "What is needed for software, then, is an engineering discipline built on the experience of software craftsman, capturing their understanding in a foundation that then can be used to educate and support a new generation of practitioners" [14]. According to Pressman, R.S. (2001), the objective of software engineering is to maintain accurate schedule and reduce cost, improve better quality products and higher productivity and all these can be achieved through effective software management, which, in turn, can be facilitated by the improved use of software metrics in software engineering [19]. According to Srinivasan, K.P., and Devi, T. (2014), "all the engineering systems are using the measure and measurement systems in day-to-day activities for the production of quality products. In case of software engineering, most of the organization produces their products without perfect

measurement system" [22] and "it is identified that software metrics research faced more difficulties towards proving usefulness in industry, theoretical validity, empirical validity, defining precise metrics, understanding, methodology of execution, execution time is more to find the metrics values, metrics are executed only by experts, and accuracy on results" [26]. The software metrics for software measurement proposed by important researchers are called C-K software metrics [8], MOOD software metrics [2], L-K metrics [16], QMOOD software metrics [5], Comprehensive software metrics (CM) [26] for object oriented design quality measurement, Halstead metrics [11], McCabes metrics [19], and Program Keyword Metrics (PKM) [22] for software coding measurement in software engineering. Recently, Srinivasan, K.P., and Devi, T. (2014), proposed a set of six Result Based Software Metrics (RBSM) suite called Comprehensive Metrics (Simple, Easy and Effective Results) for measuring Functionality, Understandability, Effectiveness, Flexibility, Extendibility, and Reusability of software design in software engineering [26]. And further, they also introduced a new kind of software metrics for software coding phase in software engineering called "Program Keyword Metrics (PKM)" [22]. This Program Keyword Metrics eliminates the important criticism called "ambiguity criticism" of most referred "Halstead Metrics" [11] and "Lines Of Coding (LOC) metrics" [19] in software coding (Program) measurement. And further they eliminated the main criticism of *"accuracy on results"* in software measurement by "Keyword Metrics (KM) (RBSM)" in Software Engineering [22]. Since 1970, the researchers of software metrics have been facing the difficulties of proving their validity using theoretical and empirical validations. There is a strong correlation between design metrics and maintainability of software system. In order to improve software design in design phase, design measurement based on software metrics is important and vital in software development. This research paper mainly concentrates on the overview of unique fundamentals of software measurement and basics of software metrics in software engineering for the improvement of the *usage of software metrics in software industries* in the following Sections. Section 2: The software measurement model of software engineering. Section 3: Characteristics of software measure. Section 4: Broad types of software measurement in software industry. Section 5: Properties of software measurement. Section 6: Principles of software measurement in software engineering. Section 7: General activities of software measurement in software engineering. Section 8: The importance of software metrics in software industry, Section 9: The characteristics of software metrics in software engineering. Section 10: History of software metrics in software engineering. Section 11: Generations of software metrics. Section 12: Types of software metrics in software engineering. Section 13: Limitations of software metrics in software industry and conclusion includes future directions of the research.

2. THE SOFTWARE MEASUREMENT MODEL OF SOFTWARE ENGINEERING

The structural model of software measurement is shown in Figure 1 describes the concepts of software measurement and their related components [15]. Formally, the software metrics require understanding of the basic concepts of software measurement activities and objects related with measurement. The structural model of software measurement is called software measurement framework. This framework describes the objects of software measurement called entities of measurement, relationships, attributes, scales and that are used for validating software metrics [15], [18]. The structural model of measurement given in Figure 1 defines an entity to possess many attributes while an attribute can qualify many different entities and it defines that an attribute can be measured in one or more units. The entities are the objects in the real world and the software measurement is to capture their characteristics and manipulate them in a formal way. For a given attribute, there is relationship of interest in the empirical world and it is to be captured formally in the mathematical world. The relationship between entities and attributes is illustrated in Figure 1. It suggests that an entity possesses several attributes, while an attribute can meet the criteria many diverse entities. A software measure plans an empirical attribute to the proper and mathematical world and a software measurement unit determines how to measure a software

attribute. Figure 1 implies that an attribute may be measured in one or more units and it implies that the same unit may be used to measure more than one attribute. Scale types are to be considered for software measurement units. The scale types are needed to understand the different measurement scale types implied by the particular unit. A unit's scale type determines the admissible transformations of a particular unit.

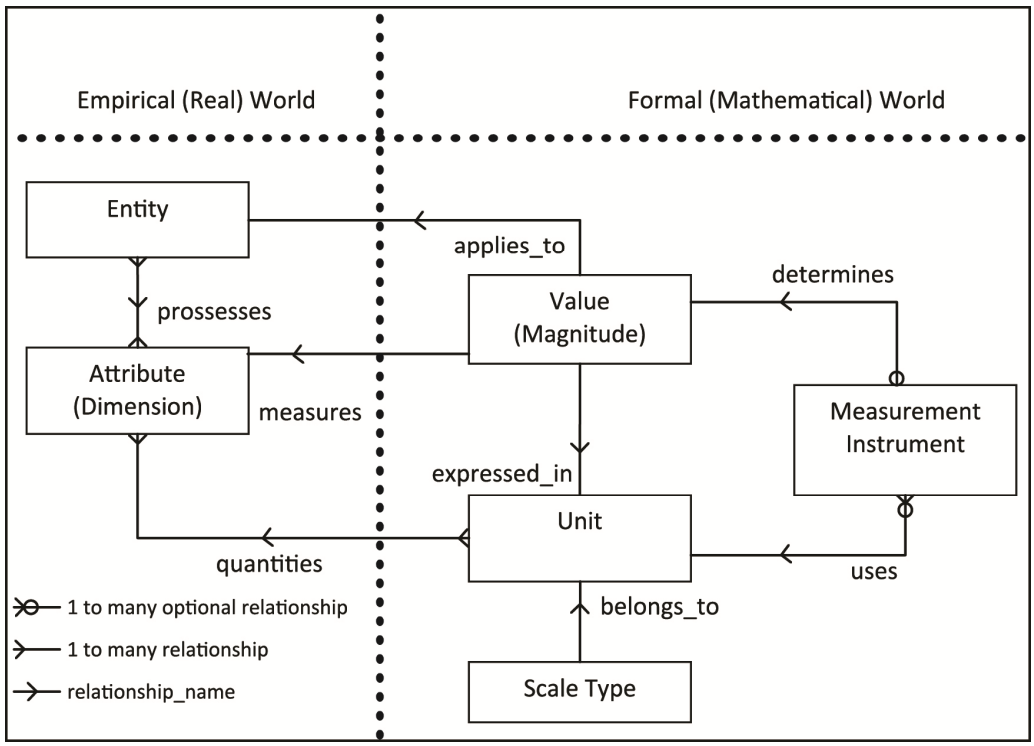

Figure 1. The Structural Software Measurement Model [15]

In traditional measurement theory, units are only applicable to ratio and interval scale measures. It is extended to the use of units in the structure model to allow for the scale points for ordinal scale measures and used for nominal scale measures. Figure 1 illustrates a one-to-one relationship between unit and scale type. In this model, different units direct to different scale types and they do not affect the attribute. In software measurement, measuring an attribute is by applying a specific measurement unit to a particular entity and attribute to obtain a value. This value is often numerical, but it does not have to be. However, these values represent a nominal scale measure and they are arbitrary labels and they cannot be summed or averaged. A measured value cannot be interpreted unless it is to know to what entity it applies to, what attribute it measures and in what unit. The software attribute has both an entity and a unit of measure. The properties of values are defined over a set of permissible values. A set of permissible values are finite or infinite, bounded or unbounded, discrete or continuous. Figure 1 shows that an instrument may optionally be used to obtain the measured value of an attribute and it indicates that there may be many different measurement instruments available for a particular unit. Measurement instruments usually detect a single unit value of an attribute in a particular unit of measurement and accumulate units into a value for a particular entity. However, instruments are also used to classify entities. In case of scalar measures that are expressed in compound units, it is usually not possible to measure the multidimensional attribute directly. There are multi-dimensional software attribute derived from several other attributes and they are measured in a compound unit constructed from relevant base units. The equation used to calculate the indirect

attribute value is derived from the nature of the multi-dimensional attribute not from any empirical association among the attributes. In the properties of indirect measures, valid indirect measures should not exhibit an unexpected discontinuity i.e., they should be defined in all reasonable or expected situations.

3. CHARACTERISTICS OF SOFTWARE MEASURE IN SOFTWARE INDUSTRY

A software measure is a numerical value computed from a set of data. In order to examine the details of software metrics, first consider the properties of a measure. The characteristics of software measure are shown in Figure 2. **• The measure should be robust.** The calculation of the software measure is repeatable and the final result is not sensitive to minor changes in environment. The software measure is precise, and the process of collecting the data for the measure is objective.**• The measure should suggest a norm, scale, and bounds.** There is a scale upon which one can make a comparison of two measures of the same type [4].

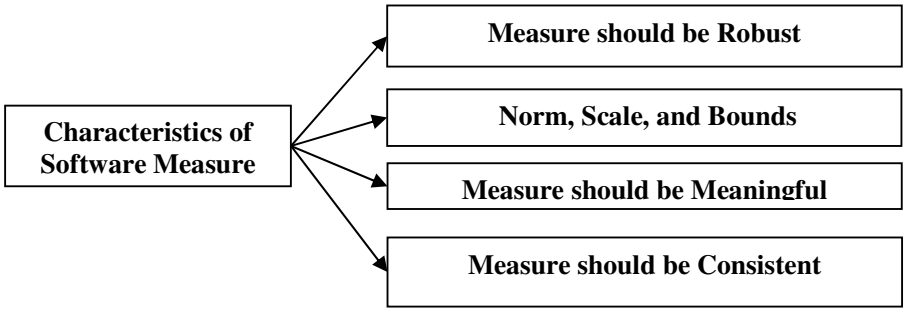

Figure 2. Characteristics of Software Measure

• The measure should be meaningful. The software measure relates to the software product, and there should be an underlying principle for gathering data for the software measure. Frequently, one measure alone is inadequate to real software measure the features of the design paradigm or to achieve the objectives of the software project in software engineering. This suggests that a suite of measures is essential to give the scope and range necessary to achieve the software project's objectives. A suite of measures adds an additional consideration.

• A suite of measures should be consistent. If a minor value is enhanced for one type of software measure in the matching set, then smaller is better for all other types of measures in the suite. In addition, the data gathering software process that produced the data from which a measure is computed should be carefully arranged.

4. BROAD TYPES OF SOFTWARE MEASUREMENT IN SOFTWARE INDUSTRY

There are two broad types of software measurement in software industry called “direct” and” indirect” software measurement methodology (Figure 3). An entity may be an object, such as a software specification, or an event. A software attribute is a characteristic or property of the entity, such as the length or functionality, or the duration of the testing. The software measurement in software engineering is defined as the software process by which numbers (or) symbols are assigned to attributes of entities in the actual world in such a way as to describe them according to obviously definite rules [9], [10], [20]. Direct measurement of a software attribute is a software measurement which does not depend on the measurement of any other attribute. Indirect measurement of an software attribute is software measurement which involves the measurement of one or more other attributes. Further, the two broad uses of measurements are shown in Figure 4 and they are**:** "assessment” and “prediction” [21]. According to measurement

theory, predictive software measurement of an software attribute A will normally depend on a mathematical model relating A to some active measures of attributes A_1 , . . , A_n. Accurate predictive software measurement is certainly dependent on careful assessment type measurement of the attributes A_1…, A_n. For predictive measurement, the model alone is not sufficient [9]. Additionally, it needs to define the procedures for determining model parameters and interpreting the results.

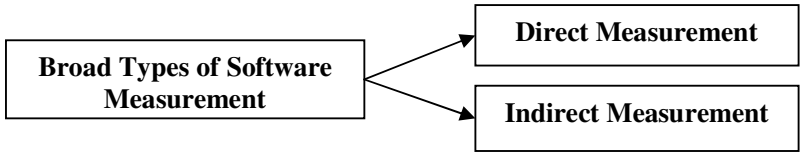

Figure 3. Types of Software Measurement Methodology

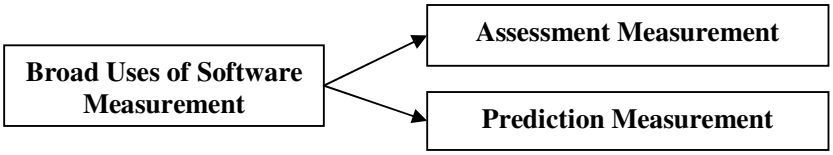

Figure 4. Uses of Software Measurements in Software Industry

5. PROPERTIES OF SOFTWARE MEASUREMENT

The concept of properties within the context of measurement theory and notation of measurement theory is called as relational system [7]. The definition of relational system, empirical relational system and formal system are defined here. The two types of relational systems are called as the empirical system and formal relational systems.

Relational System: A relational system in a measurement A is an ordered tuple (A, R_1, …, R_n, O_1, …,O_m) where A is a nonempty set of objects, and the R_i, i=1,…,n are k_i- ary relations on A and the O_j, j=1, …, m are closed binary operations.
Empirical Relational System: The empirical relational system is defined as: A = (A, R_1 ,…, R_n , o_1 ,…o_m). A = Non-empty set of empirical objects that are to be measured. $R_{i\,=}$ k_i-ary empirical relations on A with i = 1… n. $o_{j\,=}$ binary operations on the empirical objects A that are to be measured.
Formal Relational System: The formal relational system is defined as: B = (B, S_1,…,S_n , •1,…•m). B = a non-empty set of formal objects. Si = k_i -ary relations on B. •j = closed binary operations B. The relational system, empirical relational system, representation conditions, scale types are essential concepts of software measurement and software metrics [7].

6. PRINCIPLES OF SOFTWARE MEASUREMENT IN SOFTWARE ENGINEERING

The principles of software measurement are important in software metrics definitions. There are 14 principles defined for software process, software metrics and software measurement. The first four measurement principles are for the software process and the principles from 5 to 14 for the overall software measurement. The principles 5 and 6 are for the characteristics of software metrics [6]. The principles 5 to 14 are for software measurement and the descriptions of 14 principles are illustrated in Table 1.

Table 1. Principles of Software Measurement

Principles	Use / Applicable
Principles: 1-4	Software Process
Principles: 5-6	Software Metrics
Principles: 5-14	Software Measurement

Principle 1: A measurement is a perfect instrument for characterizing, evaluating, prediction, and providing inspiration for the various aspects of software construction. **Principle 2:** The measurements must be taken on both the processes and products. **Principle 3:** There is a diversity of uses for software measurement. The purpose of software measurement is obviously stated as measurements used for observing the cost, effectiveness, reliability, correctness, maintainability, and efficiency. **Principle 4:** Software measurement needs to be viewed from the suitable viewpoint. **Principle 5:** The subjective as well as objective metrics are required for software measurement. Many process, product and environment aspects can be characterised by objective metrics. Other aspects cannot be characterised objectively yet, but they can at least be categorized on a quantitative nominal scale to a reasonable degree of accuracy. **Principle 6:** In measurement, mainly aspects of processes and products are too complicated to be captured by a single metric. The definition of a set of metrics and the purpose for measurement needs to be defined. **Principle 7:** The development and maintenance environments must be prepared for measurement and analysis. Planning is required and needs to be carefully integrated into the overall software engineering process model. **Principle 8:** In general, software metrics cannot be used for other environments as defined. Because of the differences among execution models, the models and metrics must be tailored for the environment in which they will be applied and checked for validity in that environment. **Principle 9:** The measurement process must be top-down rather than bottom-up in order to define a set of operational goals, specify the appropriate metrics, and permit valid contextual interpretation and analysis. **Principle 10:** For each environment, there exists a set of metrics that provides the needed information for definition and interpretation purposes. **Principle 11:** The multiple mechanisms are needed for data collection and validation. The nature of the data to be collected from principle 5 determines the appropriate mechanisms. **Principle 12:** In order to evaluate and compare projects and to develop models needed historical experience base. This experience base should characterise the local environment. **Principle 13:** The software metrics must be associated with interpretations. **Principle 14:** The experience base should evolve from a database into knowledge base to formalise the reuse of experience. These are the basic principles for software measurement [6]. It is useful for the software measurement process, metrics and measurement. The following section explains general activities of software measurement in software engineering.

7. GENERAL ACTIVITIES OF SOFTWARE MEASUREMENT IN SOFTWARE ENGINEERING

The general activities of software measurement [13], [21] are depicted in Figure 5. As per measurement activity, first users must identify the attribute to be measured. Such an attribute must bear certain significance for a person involved in the development process. In software engineering context, a measure provides quantitative indication of the extent, amount, dimension and capacity of some attributes of a product or process.

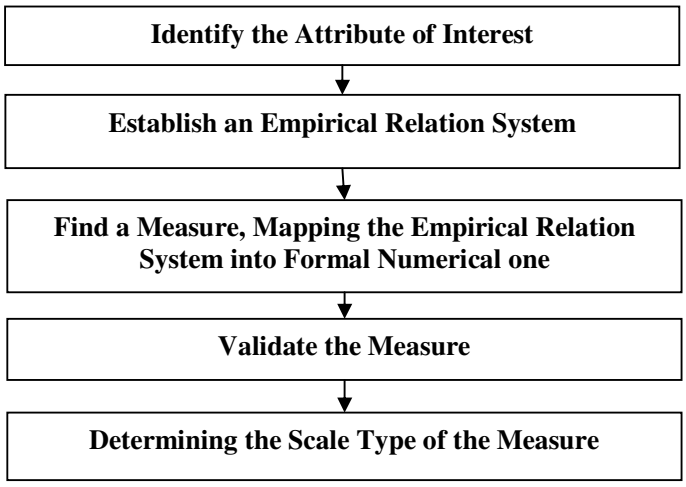

Figure 5. General Activities of Software Measurement

In the next step, an empirical relation system must be established. After that, having established an empirical relation system, a metric M should then map the empirical system into appropriate formal that is mathematical relation system. Then the next step is the task of validating a software measure in assessment sense and finally determining the scale type of measurement activity [13]. These are the activities that are essential in measurement. These activities are mostly used for software measurement construction. These activities will help to analyze and improve the measure and may guide software metrics researchers in the identification of new attributes and development of corresponding measures.

8. THE IMPORTANCE OF SOFTWARE METRICS IN SOFTWARE INDUSTRY

Almost for the past four and a half decades, software measurements have been the subject of an array of criticisms and software metrics have been proposed and given with inadequate theoretical foundations, while others have been shown to be not useful. This section explains the importance of software metrics in software industry. The software metrics are used for the development of the Process Efficiency and to improve Product Effectiveness (PEPE). The process metric is to improve the development of the software and product metric is an effort to increase its quality. Software metrics are appreciated only when (i) they are clearly defined, (ii) easy to collect, (iii) clearly understood, and (iv) it needs stand-alone metrics for measurement. In order to improve the quality and productivity of software, organizations have to integrate the measurement and process activity. Software measurement plays an increasingly important role in understanding and controlling software development practices and products [16]. Better use of existing metrics and development of improved metrics will help to achieve the goal of software engineering. According to Pressman, R.S., (Pressman, R.S., 2001) software measurement and software metrics are the key components of the software engineering discipline [19]. The software metrics are quantitative measures of a product before and after implementation. And software metrics are used to find the quality of a software process from software requirement analysis through design to implementation. Assessing the object-oriented design metrics is to predict potentially fault-prone classes and components in advance as quality indicators. In today's software development environment, object-oriented design and development is important and there is strong relationship between the object-oriented metrics and the testability efforts in object-oriented system [1, 3]. The improvement of the management software process depends upon ability to identify, measure, and control necessary parameters of the development process. This is achieved through effective

software metrics and the measurement of the essential parameters of software development. Software metrics should be used in order to improve the productivity and quality of software, because they provide critical information about reliability and maintainability of the system. In general, software metric is a measure of some property of a piece of software or its specifications. Therefore, software metrics suite is needed. Security estimation of software product must be a mandatory element of software at an early phase of development life cycle. For security estimation mechanism, there is a need to develop efficient security metrics for complexity perspective to evaluate design complexity more accurately. The recent results indicate that conscious implementation and application of software metrics can help achieve better management results both in the product and process of the software development. The detection of design defects using metrics is important for improving the quality of object-oriented software systems. By automated correction of these defects at appropriate time, total cost of software development is reduced because the manual detection of defective design is tedious and time-consuming.

9. THE CHARACTERISTICS OF SOFTWARE METRICS IN SOFTWARE ENGINEERING

There are several fundamental characteristics that are associated with software metrics in software engineering and they are given in Figure 6. The characteristics of software metrics in software engineering are simple, easy to understand; measurable, accountable, economical and precise. They must be timely, robust, independent, reliable, valid and consistent, and easily collected. The unambiguous software measurement is vital in software development process and product. The standardized software measurement, software measures and software metrics in software engineering have diverse challenges.

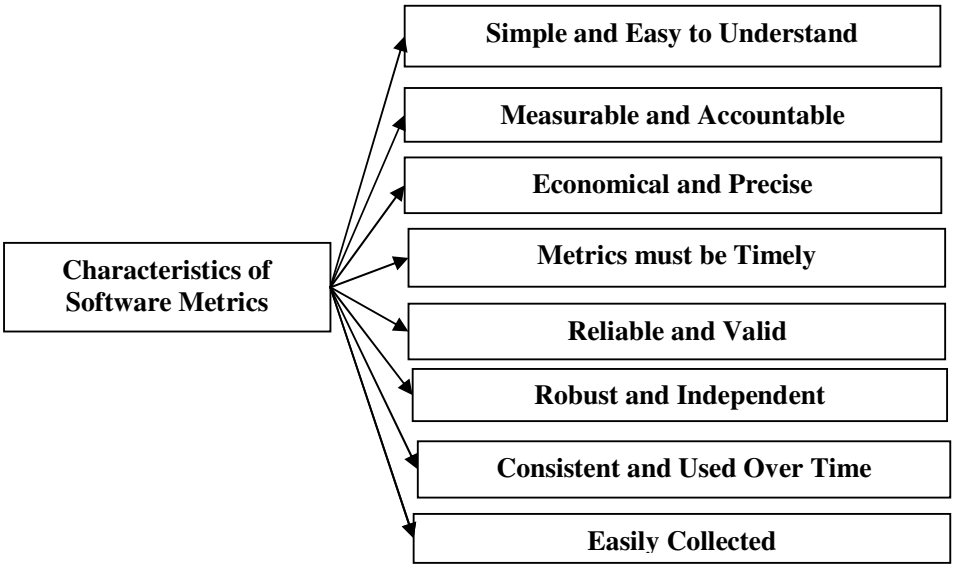

Figure 6. Characteristics of Software Metrics in Software Engineering

10. HISTORY OF SOFTWARE METRICS IN SOFTWARE ENGINEERING

The state of software metrics during the last decade is encouraging and currently, many researchers are involved in the field of software metrics. The software metrics are being applied

and good results are obtained with criticisms. Figure 7 shows main metrics milestones in history and Table 2 illustrates the main events (History) of software metrics in software engineering.

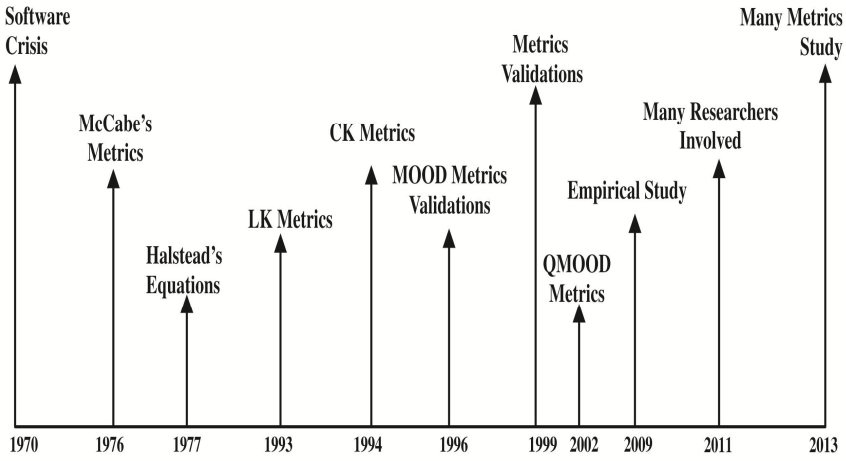

Figure 7. Milestones of Software Metrics

Table 2. Major Events of Software Metrics

Year	Major Events	Year	Major Events
1970	Software Development Crisis.	**2003**	Empirical Validations.
1976	McCabe Software Metrics.	**2004**	Cohesion Metrics - Empirical Study
1977	Halstead Software Science Equation Software Metrics.	**2005**	Empirical Validations – OOD Faulty classes' measurement.
1988	Weyuker's (Most Referred) Properties of Measures.	**2006**	Empirical Studies – New Software Metrics.
1992	Methodology for Validation.	**2007**	Empirical Validations – OOD Faulty classes measurement.
1994	C – K OOD Software Metrics. (Most Cited Software Metrics)	**2008**	Procedure Based Metrics System (PBMS) – Empirical Validations.
1995	MOOD OOD Software Metrics Validations.	**2009**	Empirical Studies – New Software Metrics.
1996	Property Based Validations.	**2010**	Empirical and new software metrics.
1997	New Coupling Software Metrics.	**2011**	Many Journal Papers Published – Total Class and System Metrics.
1998	Empirical Studies and Validity of Software Metrics.	**2012**	Many Researchers and Scholars Involved - Reviewed on Metrics.
2000	Cohesion Software Metrics.	**2013**	New Complexity, Coupling Metrics.
2001	Prediction on OOD faulty classes in software development.	**2014**	Program Keyword Metrics (PKM), Result Based Software Metrics - Comprehensive OOD Metrics.
2002	Bansiya-Davis OOD Metrics – QMOOD Methodology.	**2015**	Few Developments in Software Metrics – (Up to June 2015).

Currently, software metrics researchers have introduced novel software metrics and validated software metrics using empirical and theoretical techniques in software engineering. The present states of software metrics in software engineering has been used in decisions-making as well as in various product activities and more researchers are involved in empirical studies. The eminent researchers direct the software professionals for evaluating software product effectiveness using software metrics in software development [21-28]. The Table 2 and Table 3 shows that the current state of software metrics is still not matured based on concepts, methodology, standards, and new software metrics. At present, many researchers are involved in research on "cohesion" and "coupling" software metrics research. They are also involved in proposing software metrics for cohesion and coupling measurement. Some researchers are involved in empirical studies finding "fault-prone classes" in "object-oriented design" environments using metrics. Few researchers are involved in developing metrics tools for different environments and applied metrics tools in different applications. The main milestones and events of software metrics show that in the past history, many metrics had been proposed and validated by eminent researchers but most of the metrics lacked in experimental study and few metrics were accepted and used. Although there are many metrics in use and under active investigation, a few metrics are more difficult to apply and execute. At present, the current state of software metrics is still not satisfactory. As a result, the *battle on software metrics* is still continuing in software measurement in software industry.

11. GENERATIONS OF SOFTWARE METRICS IN SOFTWARE ENGINEERING

In the software development crisis year 1970, the software engineers emerged to focus on accurate time schedule and cost estimates, better quality software products and higher software productivity. In the software management year 1990, the software management was ineffective due to complexity of software development and software engineering had a few well-defined, reliable measures of either process or the product to guide and evaluate development. In 2015, the software metrics is used to improve the ability to identify and control essential parameters of the software development process. It must be easy to understand, execute and bring out better results of software metrics. In the result, establish the software metrics as important in software engineering for software process and product. The categorized generation of *software metrics nomenclature* [23] is shown in Figure 8. Further, Table 3 proposes the comparison of these classifications and comparative study of first and second generation software metrics.

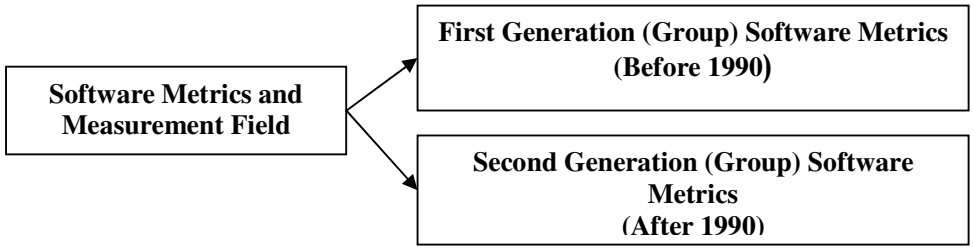

Figure 8. Generations of Software Metrics in Software Engineering

This nomenclature is proposed based on the *vast literature survey* made on software metrics. It will be useful for the researchers in order to understand and find the different stages of software metrics. At present stage, the software metrics nomenclature classification is *quite possible* for the development of software metrics field [23]. Based on the literature survey and analysis of software metrics studies, the *generations (or) groups (or) phases (or) stages* of software metrics are introduced in this research paper for betterment of future software engineering research and software industry. In wide spectrum, before 1990, the main focus of software metrics was the

complexity of the code and procedural oriented languages and after 1990, the main focus of software metrics was the object-oriented languages and software development approaches. Accordingly, the categorization of software metrics can be grouped as first generation software metrics (1970-1990) and second generation software metrics (1991-2015).

Table 3. First and Second Generations Software Metrics

Description / Study	**First Generation**	**Second Generation**
Metrics Defined by	Eminent individuals	Research Scholars and Groups
Clear Definitions	Lack of clear definitions	Defined - *Safely reaching*
Definitions for all Approach	Mainly procedural	All software development
Measuring for all Phases	Mainly coding	All phases of software
Tested	Tested by experts	Tested by any individuals
Used in Software Industry	Used by experts	Used by any individuals
Theoretical Validations	No	Few - *Safely reaching*
Empirical Studies	Very limited	Comprehensive
Experimental Study	Few	Not satisfactory
Final Results on Metrics (Accuracy on Results)	Yield different results	Result Based Software Metrics (RBSM) – PKM - *Safely reaching*
Measurement Attributes	Very limited	All attributes
Improved Quality in Software Industry	Tested	Satisfactory *Safely reaching*
Applications	Limited	All applications
Statistical Approach	Very limited	Mostly and Clear
Methodology of Execution	No	QMOOD Methodology - PBMS Methodology
Single (Total) Metrics	No	Total Class and System Metrics ((TCM and TSM)
Criticisms	More criticisms	Constructive criticisms

Illustrations in Table 3 are useful to recognise the difficulties faced by the metrics research community at each stage of four decades. Based on the illustrations given in Table 3 it is concluded that software metrics is difficult to understand and metrics execution takes more time and costly. In order to control these difficulties referred and problems faced, Srinivasan, K.P., and Devi, T., introduced the procedural based metrics system for object-oriented design quality assessment [26] and a new kind of keyword metrics for coding [22]. The Procedure Based Metrics System has been proposed for easy execution and understanding, and the execution of each step possibly reduce the confusions and gets the results for decision making.

12. TYPES OF SOFTWARE METRICS IN SOFTWARE ENGINEERING

Software plays an important role in today's life and to determine quality of the software. The *types or classification* of software metrics [21] are shown in Figure 9.

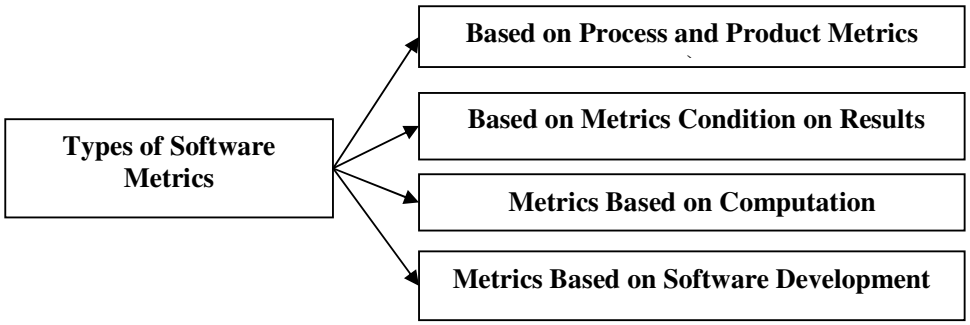

Figure 9. Types of Software Metrics in Software Engineering

The classification of software metrics is important for understanding. In general, software metrics may be broadly classified as either product metrics or process metrics and it is shown in Figure 10. The product software metrics are software measures of the product at any stage of its software development. The software process metrics are used for the measures of the software development process. The project metrics is not required in main classification because software engineering is mainly concerned with process and product and any metrics in software measurements may come under only these classifications. When the software developers use the project metrics in software engineering, such types of software metrics are called "project software metrics".

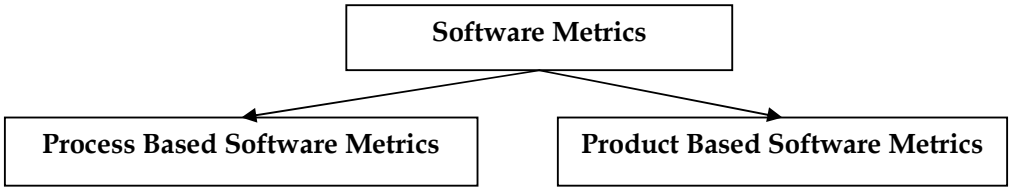

Figure 10. Types of Software Metrics Based on Process and Product

Another way of classification of software metrics is as objective and subjective software metrics and it is shown in Figure 11. A distinction is sometimes made between ``objective" and ``subjective" measures and is based on the way the measures are defined and collected. Objective software measures are defined in a totally unambiguous way, while subjective software measures may leave for interpretation. As a consequence, subjective software measures are supposed to be of lower quality than objective software measure. However, there are cases in which objective software measures cannot be composed.

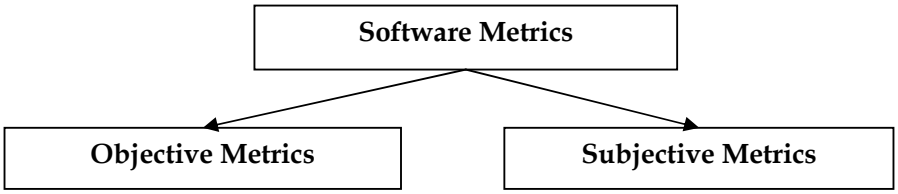

Figure 11. Types of Software Metrics Based on Metrics Condition on Results

The metrics can be categorized based on computation as "primitive metrics or computed metrics" *(Mills, E.E., 1988, SEI)* [17] and it is shown in Figure 12. "Primitive software metrics" are those that can be directly observed. "Computed software metrics" are those that cannot be directly observed but are computed in some manner from other software metrics.

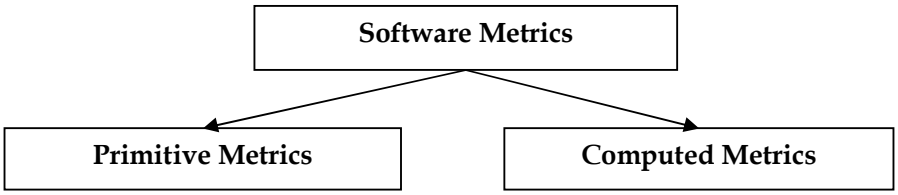

Figure 12. Types of Software Metrics Based on Computation

The software metrics can be classified based on software development model as Procedural Metrics (PM), Object-Oriented Metrics (OOM), and Web Metrics (WM) (Figure 13). The primary objective of object-oriented metrics is the same as that of the conventional software metrics. In object-oriented environment, software is a collection of discrete objects that encapsulate their data as well as the functionality to model real world called objects. Each object has attributes and method. In order to improve the object-oriented design, software measures or metrics are needed. The scales for measurement are vital in natural world. This may involve using the data in other calculations and subjecting them to statistical analyses.

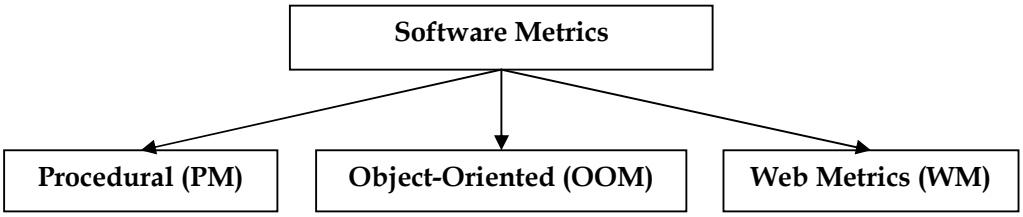

Figure 13. Types of Software Metrics Based on Software Development Model and Applications

The main objective of object-oriented metrics is to understand the quality of the product to assess the effectiveness of the process, and to improve the quality of work in a project. In literature, researchers have used these metrics names for their convenience and now it takes another form of classification of metrics. Software metrics may be broadly classified as either product software metrics or process software metrics. The classifications of software metrics are carried out by *different people by different way*. The above are the main types of software metrics used in Software Engineering Metrics (SEM) literature.

13. LIMITATIONS OF SOFTWARE METRICS IN SOFTWARE INDUSTRY

There are advantages on software measurement identified with criticisms; it can also lead to some limitations in software industry and organization. It is generally felt that the programmers are averse to software measurement and metrics and they give resistance to software measurement and use of software metrics. The main limitations of software metrics in software industry are as follows: ① The automation of software metrics is difficult task and the convention for software metrics will be difficult to implement for different environment. ② Software metric executions (at present) are time-consuming and expensive and sometimes bring out delusive conclusions. ③ It is difficult to understand software metric results, especially, if they involve in different environments and development methodologies. ④ The accurate measurement of metrics requires certified specialists on software metrics in industry.

14. CONCLUSION

The metrics are used for the development of the process efficiency and product effectiveness in software engineering. The software metrics can *help* the software professionals to make *unambiguous software attributes* of software processes and products *more visible [21-28]*. Further, measurement includes quantitative evaluations of software and usually metrics can be used directly to determine achievements of quality goals quantitatively. The current software management is ineffective due to software development which is extremely complex. An attempt has been made to concentrates on the overview [1-28] of the software measurement model of software engineering , characteristics of software measure, broad types of software measurement in software industry, properties of software measurement in software engineering, principles of software measurement in software engineering, general activities of software measurement in software engineering, the importance of software metrics in software industry, the characteristics of software metrics in software engineering, history of software metrics in software engineering, generations of software metrics in software engineering, types of software metrics in software engineering, limitations of software metrics in software industry. This research can be further extended based on concepts and methodology of software measurement and metrics in software engineering.

REFERENCES

[1] Abdullah, Khan, M.H., and Srivastava, R.,2015, "Testability Measurement Model for Object Oriented Design (TMMODD)," *International Journal of Computer Science & Information Technology,* Vol. 7, No. 1, February, pp. 153-163.

[2] Abreu, F.B., and Melo, W., 1996, "Evaluating the Impact of Object-Oriented Design on Software Quality," *Proceedings of the 3rd International Software Metrics Symposium*, IEEE, Berlin, Germany, March.

[3] Anbumani, K., and Srinivasan, K.P., 2005, "A Set of Object-Oriented Design Metrics," *Journal of The Institution of Engineers (India), IE(I), Journal – CP*, Volume 86, May, pp. 1-9.

[4] Archer C. and Stinson M.,1995, *Object-Oriented Software Measures*, Technical Report, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, April.

[5] Bansiya, J., and Davis, C.G., 2002, "Hierarchical Model for Object-Oriented Design Quality Assessment," *IEEE Transactions on Software Engineering*, Vol. 28, No. 1, January, pp. 4-17.

[6] Basili, V.R., and Rombach, H.D., 1988, "The TAME Project: Towards Improvement-Oriented Software Environment," *IEEE Transactions on Software Engineering*, Vol. 14, No. 6, pp. 758-773.

[7] Briand, L.C., Morasca, S., Basili, V.R., 1996, "Property-Based Software Engineering Measurement," *IEEE Transactions on Software Engineering*, Vol. 22, No.1, January, pp. 68-85.

[8] Chidamber, S.R., and Kemerer, C.F., 1994, "A Metrics Suite for Object-Oriented Design," *IEEE Transactions on Software Engineering*, Vol. 20, No. 6, June, pp. 476-493.

[9] Fenton, N., 1994, "Software Measurement: A Necessary Scientific Basis," *IEEE Transactions on Software Engineering*, Vol.20, No.3, March, pp. 199-206.

[10] Fenton, N.E., and Pfleeger, S.L., 2004, *Software Metrics: A Rigorous and Practical Approach*, Thomson Asia, Singapore.

[11] Halstead, M.H., 1977, *Elements of Software Science*, Elsevier, New York.

[12] Harrison, R., Counsel, S.J. and Nithi, R.V., 1998, "An Evaluation of the MOOD Set of Object-Oriented Software Metrics," *IEEE Transactions on Software Engineering*, Vol.24, No.6, June, pp.491-496.

[13] Hitz, M., and Montazeri, B., 1996, "Chidamber and Kemmerer's Metrics Suite: A Measurement Theory Perspective," *IEEE Transactions on Software Engineering*, Vol.22, No4, April, pp.267-271.

[14] Jacobson, I., and Seidewitz, E.., 2014, "Real Software Engineering," *CSI communications*, Vol. 38, Issue. 5, August, pp. 7 – 9.

[15] Kitchenham, B., Pfleeger, S.L., and Fenton, N., 1995, "Towards a Framework for Software Measurement Validation," *IEEE Transactions on Software Engineering*, Vol. 21, No.12, December, pp. 929-943.

[16] Lorenz, M., and Kidd, J., 1994, *Object-Oriented Software Metrics*, Prentice Hall, Englewood Cliffs, New Jersey.
[17] Mills, E.E., 1988, *SEI Curriculum Module SEI-CM-12-1.1*, Software Engineering Institute, December.
[18] Morasca, S.R., Briand, L.C., Basili, V.R., Weyker, E,J., and Zelkowitz, M.V., 1997, "Comments on 'Towards a Framework for Software Measurement Validation," *IEEE Transactions on Software Engineering*, Vol. 23, No.3, March, pp.187-189.
[19] Pressman, R.S., 2001, *Software Engineering a Practitioner's Approach*, 5th Edition, McGraw Hill, India.
[20] Shneiderman, B., 1980, *Software Psychology, Human Factors in Computer and Information Systems*, Winthrop Publishers, Inc, Cambridge, Massachusetts.
[21] Srinivasan, K.P., 2013, "Design and Development of a Procedure Based Metrics System for Object Oriented Design Quality Assessment," Ph. D. Thesis, School of Computer Science and Engineering, Bharathiar University, Coimbatore, India.
[22] Srinivasan, K.P., and Devi, T., 2014, "A Novel Software Metrics and Software Coding Measurement in Software Engineering," *International Journal of Advanced Research in Computer Science and Software Engineering*, Vol. 4, Issue 1, January, pp. 303-308.
[23] Srinivasan, K.P., and Devi, T., 2012, "Introducing First and Second Generations Nomenclature in Software Metrics in Software Engineering," *Proceedings of National Conference on Advances in Computer Applications*, Bharathiar University, Coimbatore, pp. 12-22. [ISBN: 978-93-80769-18-9].
[24] Srinivasan, K.P., and Devi, T., 2009, "Design and Development of a Procedure to Test the Effectiveness of Object-Oriented Design," International Journal of Engineering Research and Industrial Applications, Vol.2, No.6, pp. 15-25.
[25] Srinivasan, K.P., and Devi, T., 2011, "Design and Development of a Procedure for new Object-Oriented Design Metrics," *International Journal of Computer Applications*, Vol.24, No.8, pp. 30-35.
[26] Srinivasan, K.P., and Devi, T., 2014, "A Complete and Comprehensive Metrics Suite for Object-Oriented Design Quality Assessment," *International Journal of Software Engineering and Its Applications (Scopus Indexed Journal)*, Vol. 8, No. 2, February, pp.173-188**.** *(This Paper is recognized as "Quality Paper" by SERSC, Republic of Korea and Published Free of Cost).*
[27] Srinivasan, K.P., and Devi, T., 2014, "A Comprehensive Review and Analysis on Object-Oriented Software Metrics in Software Measurement," *International Journal on Computer Science and Engineering*, Vol. 6, No.7, July, pp.247-261.
[28] Srinivasan, K.P., and Devi, T., 2014, "Software Metrics Validations Methodologies in Software Engineering," *International Journal of Software Engineering and Applications*, Vol. 5, No. 6, November, pp.87-102. *(This Paper is recognized as "Excellent and candidate for Best Paper").*

Author

Dr. K.P. SRINIVASAN received his Master of Computer Applications Degree from Bharathiar University, Coimbatore, India in 1993. He completed his M. Phil. Degree in Computer Science from Department of Computer Science and Engineering, Bharathiar University, Coimbatore, India in 2001 and Ph. D. Degree in Computer Science from School of Computer Science and Engineering, Bharathiar University, Coimbatore, India in 2014. Presently, he is working as an Associate Professor in Computer Science in C.B.M. College (Government Aided and Co-Educational Institution), Kovaipudur, Coimbatore under Bharathiar University, Coimbatore, India since 1997. He has published five conference papers and eight journal papers. He has received the *best paper award* from a conference and "*quality paper*" and "*Excellent and candidate for Best Paper*" recognitions from a reputed journals. His current research interests are in the areas of Software Engineering and Database Systems.