Software Metric Design: Issues, Guidelines and Process

Sunil Sikka

Department of Computer Science & Engineering,
Amity University Haryana
Gurgaon, Haryana (India)
sunil.sikka@yahoo.com

Abstract

Software metric is one of the most focused areas of software engineering because of its potential benefits such as planning, prediction, monitoring, controlling, and evaluation. Lots of metrics are available in literature out of which only few are universally accepted by software industries due to four reasons: 1) Lack of explicitly well defined Goal/Objective 2) Undefined Context 3) Lack of Validation 4) Inconsistency. To overcome these problems the metric design process must deals with all these issues. This paper addresses all these issues and discusses few guidelines studied from literature to develop useful metrics. The metric design process is also presented to deal with these issues.

Keywords-Software Metric; Metric Design Process; Formalization

I. INTRODUCTION

Software measurements play very significant role in software engineering. Importance of measurements is collectively recognized by researcher's community, software industries and software standards. Metrics are used for measurement in software development and management, to provide quantitative and objective base to software engineering for decision making. Software metrics can help to address the most critical issues in software development and provide support for planning, predicting, monitoring, controlling, and evaluating the quality of both software products and processes [1]. Software metric is a collective term used to describe the very wide range of activities concerned with measurement in software engineering [2]. Appropriately selected metrics can help both management and engineers to maintain their focus on goals [3]. Monitoring progress of software for taking corrective actions, evaluating product and process, planning resource and time, predicting size, quality or other attributes of the delivered software are some important uses of metrics. Along with many advantages metrics also suffers from some critical issues such as lack of explicit and well defined goals, undefined context, lack of validation, and Inconsistency. To deal with these issues it is necessary to consider all these issues during metric design process. This paper addresses all these issues and provides some guidelines collected from literature. These guidelines should be followed to produce fruitful metrics. A context aware formal metric development process is also presented.

II. ISSUES RELATED TO METRIC

There are four major problems related to software metrics that makes them ineffective. It is significant to understand these problems and their causes/sources before designing any metric. The metric development process must resolve all these issues. Following are the four problems:

A) Lack of explicit and well defined goals

The first major problem with metrics is inadequately defined goals. Metrics are not always defined in the context of some explicit and well defined measurement goal derived from an objective of industrial interest [4]. This is one of the major reason due to which metrics proposed by various researchers are not accepted by industries. Therefore without well-defined goals, metric is useless. Following are the few causes of this problem

- Too generic goal for example the goal "To reduce complexity of software" is not well defined goal.
- Sub goals are not defined.
- Goals are not derived from industrial interest.
- Goal specification is not formalized.
- Criteria of goal achievement are not specified.

B) Undefined context

Metric specification may lack the precise context specification i.e. environment in which they will be used. When context of a metric is not defined then it may be misinterpreted by its users. For example applying component oriented metrics in object oriented environment.

C) Lack of validation

It is necessary to validate the metrics before implementing it. But most of the metrics suffers from lack of validation. Without validation it not sure that metric is measuring the same for which it is developed and correlated with existing metrics.

D) Inconsistency

Another major problem with metrics is inconsistency. Many metrics are not uniformly interpreted by all users. Inconsistency is the major issue in software measurement right for the beginning i.e. Line of Code (LOC) counting. Now when software engineering is so mature even then we are searching for a consistent terminology of measurement. For instance, software measurement researchers and practitioners have not reached an agreement on the precise meaning of some terms commonly used such as 'measurement', 'measure', 'metric', 'measurable attribute' etc. [5].

III. GUIDELINES FOR DEVELOPING METRICS

Following are some guidelines derived from literature for developing useful metrics.

A) Metric must have use and user

Obviously if some metric is being developed it must have some user who will use this metric and benefited by its feedback. The user may be project manager or developer or tester or any person involved in software management and development. The uses of the metric also must be clearly specified which guides the user why and what metrics should be used to get its maximum benefits.

B) Formalization

Formalization applies to followings

C) Formally specify the goals

The basic definitions of measurement suggest that any measurement activity must proceed with very clear objectives or goals [6]. One of the reasons due to which many metrics are not accepted by industries is lack of explicit and well-defined measurement goal derived from an objective of industrial interest [4]. Therefore to develop a metric we must have a goal which is derived from an objective of industrial interest. There must be some formal method to specify the goal.

D) Formally specify the metric

A metric should be unambiguously defined i.e. metric should be interpreted in same way by its all users. One way of doing so is to define precisely what mathematical properties characterize these concepts, regardless of the specific software artifacts to which these concepts are applied [7]. Some non mathematical methods are also available to formally specify the metrics. For example Object Constraint Language (OCL) can used to define object-oriented metrics formally.

E) Follow formal method to map goals to metric

The approach used to find metrics to fulfill the specified goals also must be formally specified.

F) Metric should be cost effective

Metric cost includes cost of data collection, applying metrics, training and change in software development process etc. Benefits include increase in quality, predictable software process and better decision making. It is hard to quantify the benefits of metrics therefore heuristics may be used for cost benefit analysis. If cost exceeds the benefits then some alternates should be considered to reduce its cost. If it is not possible to reduce the cost then management should decide whether to use the metric or not.

G) Metric should be applicable as early as possible

It is important to find out the artifact(s) required to compute the metric. It is widely recognized that the production of better software requires the improvement of the early development phases and the artifacts they produce [8]. Earlier the required artifact is available during software development earlier the metric can be applied. The more early a metric can be applied in software development more it is advantages. For example computing quality of software by designs is more advantages as compare to using test case. Briand et. al. [8] emphasis on the early availability of significant metrics for better software development and management process because it allows for early detection of problem, better software quality monitoring, quantitative comparison of techniques, empirical refinement of process and more accurate planning of resources.

H) Environment and assumptions must be specified

Development of metrics must consider the environment or context in which they will be applied. Metrics must be driven by Context's (process, problem domain, environmental factors etc.) characteristics in which metric will be used [8]. The identification of universally valid and applicable measures may be ideal, long term research goal, which cannot be achieved in the near future , if at all [4]. Also any assumption about metric must be specified.

For example if an object-oriented metric of reusability is based on single inheritance only then it should be specified that metric is not applicable in case of multiple inheritance.

I) Easy to compute

One of the good characteristics of metric is, it should be simple to compute. If the metric is just base measure then it is very easy to compute, because it can be directly computed. For example total number of sub classes in object-oriented design. Derived metric is a function of two or more variables. For example Lack of Cohesion in Methods (LCOM) is a derived metric.

J) Apply measurement program

Applying metrics manually on large and complex software is not only time consuming but also cumbersome and error prone. Therefore, methods are required to compute metrics automatically. Metrics can be computed automatically using specially designed measurement programs.

K) Side Effects must be identified and controlled

It is essential to identify that how metric can be misused in any organization and the side effects of metric. The unintended side effects may be slowing rather than streamlining the organization, and can even serve to obscure our understanding of test results and reduce the overall product quality [9]. Hoffman [9] and Kaner et. al. [10] provides some side effects due to the measurements. Measurements can change the employee's behavior in order to make the measurements look better artificially rather than reflecting the actual status of attributes by ignoring the actual goals of organization. Kaner et. al.[10] defines a measurement system yields distortion if it creates incentives for the employee to allocate his time so as to make the measurements look better rather than to optimize for achieving the organization's actual goals for his work. There must be some control strategy to avoid these side effects. Organization should motivate employees to produce actual results of measurements rather than criticizing any employee based on the measurements results.

L) Use validated metrics

Validation of metric is necessary for successful software measurement because non validated metric can be misapplied i.e. metric can be used that have no relevance to the property being measured. IEEE Std. 1061-1998[11] defines validated metric as a metric whose values have been statistically associated with corresponding quality factor values. This definition is particularly for quality metric but same is true for every type of metrics. According to [12] a measure is valid if it actually measures what it purports to measure and it is useful i.e. it is related to some external attributes worth studying and therefore helps reach some goal(e.g. assessment, prediction).IEEE Std. 1061-1998[11] emphasize on the use of validated metrics only i.e. direct metrics or metrics validated with respect to the direct metrics. A metric validated in one environment need not necessarily be valid in other environments or future applications. Therefore metric shall be revalidated before it is used in other environment or application. According to Fenton[6] valid measures in assessment sense must follows representational condition i.e. there must be some mapping which maps an empirical relation system to numerical relation system in such a way that empirical relations are preserved. For validation of measures in predictive sense all the components of measure and hypothesis must be properly specified before starting validation. Empirical and Analytical validation are two types of validation techniques. Empirical validation shows that metric being validated is correlated with existing metrics. Valid metric must have high degree of association with existing metrics. It is a data based validation technique coming with conclusions which is verified. For empirical validation it is necessary to specify in advance the experimental hypothesis and dependent variables. Analytical validation is a theoretical validation technique which validates the measures by predefined properties or models. This type of validation is concerned with demonstrating that a measure is measuring the concept it is purporting to measure [12]. Kaner et. al. [10] provided a framework for proposed metric evaluation to solve the question "How do you know that you are measuring what you think you are measuring" the evaluation framework consists of ten questions which must be answered. Weyuker[13] propose nine properties for analytical validation of complexity measures. Weyuker properties are criticized by many authors. According to Fenton[6] the main drawback of Weyuker axioms is that they try to validate all types of complexities by same properties. It is impossible to measure all type of complexities using a single measure. Fenton proves that we can not even measure control flow complexity using single measure therefore general complexity measure is impossible. Fenton proves in informal way that Weyuker properties are incompatible because a "size" type complexity measure should satisfy property -5 but "comprehensibility" type complexity measure cannot satisfy property-5. On the other hand property-6 has much to do with comprehensibility and little to do with size. Briand et. al. [7] propose generic properties of Size, Length, Complexity, Cohesion and Coupling measures for theoretical validation.

Xu Jie et al. [14] addressed an important research question that "How effective software metrics can be validated in managing software quality control and estimation?". Authors discussed correlation analysis, Analysis of Variance (ANOVA) and machine learning techniques for metric validation.

IV. METRIC DESIGN PROCESS

The metric design process consists of pre process activities, inputs, sub processes (mapping goal to metric and validation) and post process as shown in figure-1. To deal with all issues discussed in section-2 it is the essential to formally specify all the inputs, outputs and process required to develop metrics.

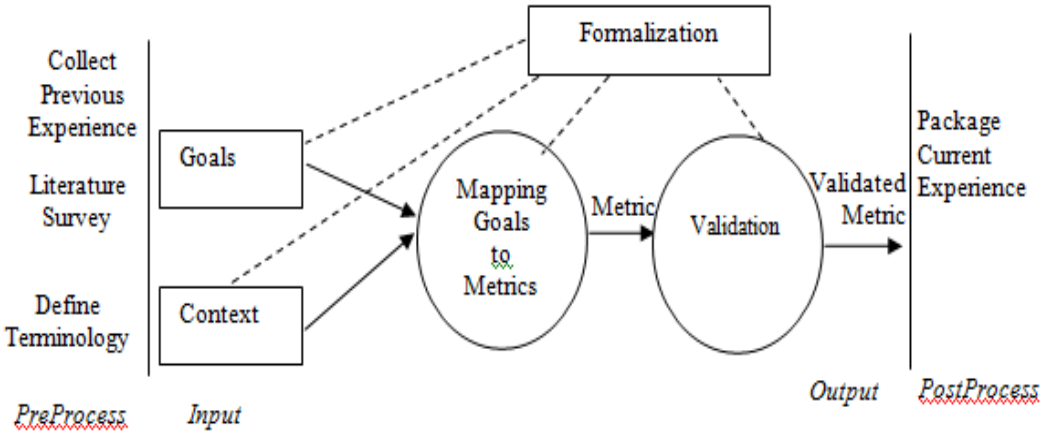

Figure 1. Metric Development Process

Formally specified metrics are internally consistent as well as uniformly interpreted by all its users. Formally specified goal and procedures generates valid metrics and also helpful for their automation.

The pre process activities include collection of related previous experiences of the organization, literature survey and definition of measurement terminology used in the process. The previous related work is very import to collect before designing any new metric. It provides the organization with a variety of information relevant to the way an organization develops software, e.g. quantitative prediction models, lessons learned from past projects, measurement tools and procedures, or even raw project data[4].

Goal(s) and Context are two inputs in metric development process. The basis theory of measurement also state that any metric development must be initiated by one or more than one goals. The goals should be defined in some formal way. For example Goal/Question/Metric (GQM) paradigm defines a template for specifying the goal. By fulfilling values of various parameters, we can specify any goal. Five parameters of this template are object of interest, a purpose, a quality focus, a viewpoint and description of environment. The general format of template is "Analyze ……for the purpose of…….with respect to…….from the viewpoint of the ……..… in the context of ……". For successful metric generation the environment for which metric is to be developed must be precisely defined. The environment determines the scope in which the results of the study are valid and can be generalized [4].

To convert goals into metrics one of the well-known frameworks is GQM which finds the necessary metrics to fulfill the goals through questions which itself is elicited from goals. Validation process validated the metric using some pre defined properties of metrics or using data based approach. Finally experiences of current work are packaged as a post process activity.

V. CONCLUSION AND FUTURE DIRECTIONS

The issues addressed in this paper highlight the problems that must be resolved during new metric design. The guidelines discussed in this paper are useful for developing fruitful metrics. Metric development process provides systematic way to develop metrics. The process yields formally specified consistent and validated metrics. This paper throws some light on the important issues addressed in literature. However, more detailed study need to be conducted for metric design.

REFERENCES

[1] V.R.Basili , H.D.Rombach "The TAME Project: Towards Improvement-Oriented Software Environments" IEEE Transactions on Software Engineering, November 1988.
[2] Norman E.Fenton "Software Metrics: Roadmap" Proceedings of the Conference on The Future of Software Engineering, 2000 Limerick, Ireland pp 357 – 370. http://www.dcs.qmul.ac.uk/~norman/papers/metrics_roadmap.pdf
[3] Linda Westfall "12 Steps to Useful Software Metrics" www.westfallteam.com/Papers/12_steps_paper.pdf
[4] Lionel C. Briand, Sandro Morasca, Victor R. Basili "An Operational Process for Goal-Driven Definition of Measures" IEEE Transactions on Software engineering, Vol. 28, No. 12, December 2002.
[5] Felix Garcıa, Manuel F. Bertoa, Coral Calero, Antonio Vallecillo, Francisco Ruı´z, Mario Piattini, Marcela Genero "Towards a consistent terminology for software measurement" Information and Software Technology 48 (2006) 631–644.
[6] N. Fenton "Software Measurement: A Necessary Scientific Basis" IEEE Transaction Software Engineering, 20, 1994, pp. 199-206.
[7] L. Briand, S. Morasca, V. Basili "Property-Based Software Engineering Measurement" IEEE Transactions on Software Engineering 22(1), 1996, pp. 68-85.

[8] L. Briand, S. Morasca, V. Basili “Defining and Validating High-Level Design Metrics” Technical Report CS TR-3301, University of Maryland, 1994.
[9] Douglas Hoffman “The Darker Side of Metrics” presented at Pacific Northwest Software Quality Conference, Portland, Oregon, 2000. www.softwarequalitymethods.com/Papers/DarkMets%20Paper.pdf
[10] Cem Kaner, Walter P.Bond Software Engineering Metrics: What Do They Measure and How Do We Know? 10th International Software Metrics Symposium, Metrics 2004
[11] IEEE, "IEEE Std. 1061-1998, Standard for a Software Quality Metrics Methodology, revision of IEEE Std. 1061-1992." Dec. 1998.
[12] Lionel Briand, Khaled El Emam, Sandro Morasca “Theoretical and Empirical Validation of Software Product Measures” International Software Engineering Research Network Technical Report #ISERN-95-03.
[13] E. Weyuker “Evaluating Software Complexity Measures” IEEE Transaction on Software Engineering, 14(9), 1988, pp. 1357-1365.
[14] Xu Jie., Ho Danny. and Capretz Luiz Fernando (2010) "An empirical study on the procedure to derive software quality estimation models", International Journal of Computer Science & Information Technology (IJCSIT), AIRCC Digital Library, Vol. 2, Number 4, pp. 1-16.