



Software Metrics Usage & Research - Gaps and Future

¹Mandeep Walia, ²Anu Gupta, ³Ravinder Kumar Singla

¹PhD Research Scholar, ²Associate Professor, ³Professor

Dept. of Comp. Sc. & Applications

Panjab University, Chandigarh, India

Abstract: Several organizations have institutionalized comprehensive metrics programs to strengthen management decision making and enable continuous improvement of software engineering processes. A range of software metrics focusing on various quality attributes and abstraction levels have been recommended by the research community. Recent studies indicate, however, that almost 80% of the software metrics initiatives do not succeed. The study depicted in this paper is a reflection of the current state of adoption of software metrics in industry vis-à-vis research going in academia and highlights areas for further improvement and future research. The journey ahead for software metrics lies in using relatively simple existing metrics to formulate management decision-support systems that associate different aspects of software development and testing processes and equip managers to make many types of evaluations, predictions and trade-offs during the entire software life-cycle. The approach suggested in literature is to handle the key considerations largely overlooked in the usual metrics models namely: causality, ambiguity and different associated subjective issues. Therefore, the way ahead for research on software metrics lies in causal modelling, empirical software engineering, and multi-measure decision aids. This paper is an outcome of a thorough literature review exercise carried out on manuscripts, articles and findings related to software metrics presented over past four plus decades accompanied by the experience and observation of authors in the domains of industry and academia. The research is hence aimed at both scientific and practicing communities.

Index Terms: Software Metrics, Measurement, Software Management, Software Process Improvement, Software Testing, Outsourcing

I. INTRODUCTION

The field of Software metrics is over 40 years old, but it has hardly been able to penetrate into the mainstream software engineering. A key cause for this has been that most software metric activities have not addressed their most important objective: to provide adequate information to support quantitative managerial decision-making during the software engineering lifecycle. Reliable support for managerial decision-making means support for risk evaluation and reduction. Yet conventional metrics approaches, generally based on regression based approaches for cost estimation and defect forecasting, provide inadequate support for managers desiring to use measurement towards analyzing and minimizing risk.

Software metrics came in to existence almost the same time as software engineering. However, the gamut of literature produced related to the field and the research work carried out could hardly impact organizational arena in a remarkable way. This is not very encouraging since the catalyst factor for metrics adoption has been to enhance managerial decision making process. The metrics activity happening in industrial context has been mainly centered on the metrics, which came in to existence 4 plus decades ago (viz. size, defect counts, LOC). No doubt, these basic metrics can be termed as winning ones looking at their wide-spread usage and acceptability, but there misapplication is also very common. A key issue with these metrics is that they generally work in an isolated fashion with a primary focus on their underlying definition and ignoring various other associated aspects. It is fairly practical and possible to come up with management decision making systems which are based on simple metrics like these, but also take in to consideration the environmental and other associated perspectives.

Of late software metrics have been most often recommended as quantitative tools of choice in the arena of software engineering, and their analysis has been most often discussed in 'measurement theory'. However, in other subject areas, it is the field of knowledge referred to as 'metrology' that is considered as foundation for development and usage of measurement instruments and associated processes.

Software measurement has been a hot topic at industry conferences in recent years, with presentations ranging from advanced statistical process control models to simple models of project effort/cost tracking. Partly, due to such motivating factors as the Software Engineering Institute's Capability Maturity Model Integrated (CMMI®), more and more organizations have adopted software metrics for managing their software development

programs. The most enthusiastic supporters endorse measurement as a "silver bullet" and a remedy for everything that affects the software industry, while critics refuse to even admit that benefits are achievable "only" with adoption of software metrics. Several flavors of software metrics exist along with many reasons for measuring. The saying: "you can't manage what you can't measure" has become the motto for many software measurement initiatives - many organizations seeking to enhance management and control of their software engineering processes through measurement.

Grady [34] describes both tactical and strategic objectives for adoption of measurement. As per Grady, "project managers must define the right product, execute the project effectively, and release the product at the right time. Software metrics help to clarify those details," and later, "As more people and organizations have adopted metrics, metric usage has evolved to become a strategic advantage" (through software process improvement). Largely, a well-planned and implemented metrics program enables an organization to recognize, standardize, advance and leverage its software engineering best practices.

Hitherto, in spite of these published successes, almost 80% of software measurement programs die down within the first 2 years [18] of their inception. This elevated failure rate has been consistent over the past couple of decades, and has taken place in spite of ever-increasing industry expertise and various guiding models on how to establish software metrics programs. A survey of on hand literature on software measurement uncovers that there is a tremendous focus on the technical facets of measurement, but really thin on the cultural and human aspect of metrics implementation.

Software project teams have wide-ranging metrics and measurement models at their disposal to monitor vital attributes of the software at hand, associated development and testing processes as well as the team itself. Sorry to say, in actual software project teams make little effective use of software metrics to ensure success.

A considerable gap exists between theory and practice due to a lack of viewpoint in the research consideration given to the subject area. In the past few decades, a number of software metrics and measurement models have been devised. However, their validation is usually restricted to establishing the construct validity of a metric. Although crucial, the time has arrived to go beyond this type of validation and examine usefulness of software metrics in specific situations accompanied by human behaviors.

All these evidences support the inference that there is more to be done to institutionalize a successful measurement program that meets the eye. The secrets of winning metrics programs outspread well beyond the technical precision of preferred metrics and collection of associated data.

II. WHY METRICS?

Software metrics have been an essential part of the state-of-the-practice in software engineering. More and more customers specify efficiency and/or effectiveness metrics reporting as part of contractual requirements. Industry standards viz. ISO 9000 and industry models viz. Software Engineering Institute's (SEI) Capability Maturity Model Integrated (CMMI®) have a special focus on measurement. Software companies are using metrics to better assess, track, control and forecast software projects, underlying processes and end-products. The term of software metrics implies different things for different people. When we purchase a book or read an article on software metrics, the focus can range from project costs and effort prediction/modelling, to tracking defects and associated Root Cause Analysis (RCA), to certain test coverage metrics, to computer performance indicators. All these are instances of metrics when the word "metrics" is used as a noun.

According to the activity oriented view envisaged by Goodman [33], software metrics is, "The continuous application of measurement-based techniques to the software development process and its products to supply meaningful and timely management information, together with the use of those techniques to improve that process and its products

III. HISTORICAL VIEW OF SOFTWARE METRICS

To have a reasonable assessment of the current state of software metrics and how they have fared in the past, we should have a quick glance at their past. The first book related to the topic got published in 1976 [51] and the concept got materialized in 1960's. The very first metric, which went in to actual usage was LOC. At that time, it was being used as a proxy measure to gauge engineer productivity (LOC/person month) or software application quality (defects/KLOC). Initial resource prediction methods also leveraged LOC or other similar measures as the main size variables. Back in the year 1971, Akiyama [4] recommended a basic regression-oriented model for measuring module defect density as defects per KLOC. This too was equivalent to using KLOC as a proxy measure for gauging program quality. This has been considered as the first attempt to utilize metrics for predicting software quality.

The clear-cut disadvantages of this proxy measurement approach with underlying crude measures like LOC were acknowledged during 1970's itself. Due to the basic flaws in the LOC measure itself viz. a line of code in a high-level programming language being drastically different from the one in an assembly-based program in terms of effort and complexity, led to the realization of the need for more foolproof and differentiating measures.

This led to the invention of programming language independent and more versatile measures of software complexity and size like Function Points back in late 1970's. Post that started subsequent endeavors in academia and industry of enhancing these measures to be applicable for evolving trends of the OO-Object Orientation and beyond.

IV. CURRENT STATE OF METRICS ADOPTION AND GAPS

If we try to gauge the advancement in the field of software metrics in terms of no. of published books, articles, journals and conferences, then the achievements have been remarkable. The amount of papers published in the field has been growing at an exponential rate. There are thousands of books available on software metrics, perhaps as there would be on software engineering as a subject area. Most of the software engineering courses running universally now comprise of some essential focus on software metrics. This indicates that software metrics as a subject area has run hand-in-hand with the area of software engineering.

If we observe last 15-20 years, there has been a tremendous surge in the extent of software metrics adoption and practice across the IT industry. Most of software organizations ranging from medium to large sized ones have adopted formalized software metrics endeavors.

Measurement, metrics and statistical analysis of data have been regarded as the basic tools of science and engineering including software engineering. Unfortunately, the software industry has been in existence for over 50 years with metrics that have seldom been properly justified, and with statistical procedures that are openly questionable [42].

There is poor alignment in the essence of software metrics action and endeavors happening in academia and industry. Similar to various other key areas, which are regarded as core of the software engineering e.g. formal methods, object-oriented analysis and design etc., the research work being carried out in academia has rarely been accepted and adopted by the industry. One notices that even today the industrial metrics activity is centered on metrics, which came in to existence back in 1970s.

For many years, software metrics have been successfully employed to assess, monitor, and improve software engineering products and processes. There is bulk of literature available portraying successes and methods of software metrics and organizations regularly implement varied kind of metrics programs. In software engineering and quality assurance there is almost blind agreement on the value of such programs, although in many of these organizations, the implemented metrics program is slowly leading to lower productivity and quality.

It is imperative for any organization concerned about quality of their products and services to be aware and wary of metrics. Even organizations having well established metrics programs, particularly organizations with long running metrics programs, should consider whether the metrics in adoption have the desired meaning and recognize what side effects are produced. Where efforts are invested without being able to improve the process and product quality, some enquiry should be made as to the aptness of measures and metrics in use. The unintentional side effects might be slowing rather than improving the organization, and can even aid in complicating our understanding of observed test results and reduce overall product quality.

Given the well-recognized value of software metrics in industrial arena accompanied with constant research going on in academia, we observe below key aspects of deficiencies in software metrics maturity, which need to be addressed by further research. Further, this research needs to be pursued in a setting wherein there is a close collaboration amongst industry and academia. All these deficiency aspects have been synthesized in Figure-1 below:

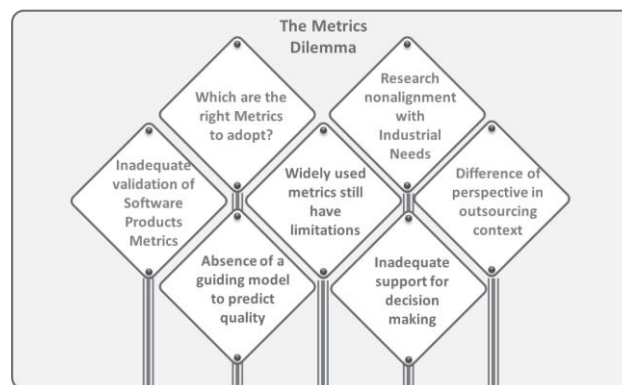


Figure-1. The Metrics Dilemma

A. Which are the right Metrics to adopt?

Unquestionably, metrics hold a fairly high potential in improving software engineering processes, but there are

so many metrics to choose from. Project managers hardly have time to make a fitment analysis of a gamut of metrics and then choose the “right” ones, which are apt for their projects. Since every project is different, is executed in a different environmental context and has its own specific set of associated problems to be tackled, even the rich set of examples and illustrations provided in literature do not serve as sufficient guiding tools for the decision making community.

Whereas the literature acknowledges that metrics are an important foundation for control in software engineering processes and products, project managers still ask the question on the most suitable set of metrics for adoption in their projects. There are hundreds and thousands of metrics available, however there is a dearth of guiding aids, which can help project managers in making the right metrics set selection based on their respective project situations.

Some of the major drawbacks in the current way metrics are institutionalized are their inter-mixing with evidences, usage in isolation and adopting inadequate set of metrics. Adopting numbers only to make your management style quantitative does not imply that the measurements are being leveraged in a meaningful and reliable way to reap the intended benefits. Such an adoption without thorough considerations is considered as under-utilization of metrics. The finest example here is of LOC metric, which could give confusing views of project attributes viz. size, cost and progress.

For many of the available metrics, the evaluation comprises of confirming the mathematical attributes of the metric, examining the metric’s behavior for a number of open source software systems or matching the value of the metric with other similar metrics quantifying associated quality attributes. Unfortunately, a structural investigation of the practicality of metrics in a real-world setting is often omitted. Such an investigation is important to understand situations in which a specific metric should be applied, to recognize areas of potential improvements, to discover general problems uncovered by these metrics and to lay down generally applicable solution approaches.

On the whole, we observe that several concepts and best practices related to software metrics have been recommended. Yet, a major drawback remains: it is still unclear which metrics to use. There are no recommendation models, which project managers can use as their guiding models, which help them make the right selection of that most appropriate metrics set for their specific needs. In spite of the considerable effort that went in to progressing research in the area of software metrics, there is scarcity of a consolidated view of these findings, which practitioners can back upon as guiding means to understand the what, when and how aspects of adopting measurements and metrics.

B. Academic Research nonalignment with Industrial Needs

Lot of research related to software metrics has been happening in academia over the years with researchers trying to identify varied attributes of software processes and products, and devising/recommending metrics around those attributes. Most of the academic metrics research however has been fundamentally irrelevant to industrial needs. Here, Irrelevance can be segregated in to two levels:

1. Scope Irrelevance: most of the academic work has been oriented to metrics, which are more amenable for application in small-scale projects, whereas the real essence and benefits of application of metrics can be realized in relatively large sized enterprise kind of software projects. Many of the metrics models developed in academia are built on attributes and parameters, which are not even realistically possible to measure and this difference adds to the irrelevance in scope dimension.

2. Content Irrelevance: there has been a marked misalignment in the attributes under focus for metrics definition and association amongst industry and academia. Whilst most of the enterprise-level metrics programs have been focusing on software process improvement, much of the academic efforts and research has been concentrated on software applications, their design and coding aspects. Academic focus has been severely lacking on the process front as if the know-how and essence of software engineering processes lies solely with the software practitioner community. In fact, such kind of irrelevance impelled Glass to remark:

“What theory is doing with respect to measurement of software work and what practice is doing are on two different planes, planes that are shifting in different directions” [32]

Even in industrial arena, much of the metrics movement and adoption seems to be more of enforced kind of rather than taken with a real passion of implementing the underlying themes of enhanced assessment and continuous improvement, prevention over control and assurance of process and product quality. Most of the metrics initiatives launched by organizations are more of top-down enforcement and are not taken very positively by the middle management and engineer communities. Moreover, such initiatives would get launched only when there are some real issues to tackle with, as a corrective measure rather than something taken as a basic ingredient of your organization’s way of working. Another trigger factor for such initiatives would be the organization’s binding to adhere to certain external standards and compliance models viz. the Capability Maturity Model Integrated (CMMI®). Since attaining higher levels of CMMI® demand organizations to adopt metrics in certain way, they would be bound to do that.

While most of the operational and mid-level staff see metrics adoption as an overhead to their jobs, senior management have their own focus and intent of institutionalizing metrics programs. Such programs, while from

top look like getting executed smoothly and giving some direction, most of these directions look to be incomplete and inconsequential. You will not find many success stories related to software metrics payback that can be replicated in similar environmental contexts. When there are delivery pressures due to tightened budgets and/or schedules, activities related to metrics are generally the first ones to suffer. Hence, while management have their own view and expectation of deploying metrics, unless there is passion and dedicated adoption from the actual execution level staff, metrics programs would rarely be successful.

You will encounter numerous instances of industrial metrics practices, where the proven principles of data collection and analysis will get neglected hence applying metrics in ways which are not considered valid. For instance, some organization trying to gather defects data and not distinguishing amongst whether the defects got detected in production or during development. Such instances of “metrics misuse”, when practiced would obviously not yield expected results.

C. Inadequate validation of Software Products Metrics

From an Industrial adoption perspective, though the process facet of software engineering has been seen a good amount of focus, the software product aspect has seen relatively more recent traction with the advent of enterprise products.

A decent number of metrics for software products have been proposed in the software engineering literature. Product metrics help us to quantitatively characterize some characteristic of the structure of a software product viz. a requirement specification, design or source code. These are also frequently collectively termed as “complexity metrics”. Whereas many of these metrics have been based on upright ideas about what is significant to measure in software products to capture their complexity, it is nevertheless essential to systematically validate them. The modern software engineering literature has indicated a key concern on the value of methods, which are used to validate software product metrics. The concern raised is driven, at least partly, by a recognition that: (a) most popular practices used for validation of software metrics are not agreeable on scientific grounds, and (b) valid metrics are crucial for effective software project management and comprehensive empirical research. For instance, in a recent paper [46], the authors mention: "Unless the software measurement community can agree on a valid, consistent, and comprehensive theory of measurement validation, we have no scientific basis for the discipline of software measurement, a situation potentially disastrous for both practice and research." Hence, in order to have a sense of confidence in the usefulness of many metrics that are invented/proposed in research labs, it is critical that they are validated thoroughly.

In case of software products, such a validation would mean realistically demonstrating that:

1. The metric measures what it purposes to measure e.g. that a coupling metric is actually measuring coupling
2. The metric is related to some significant external metric (viz. metrics, which measure maintainability and reliability)
3. The metric is an improvement over current set of product metrics e.g. a new metrics that is easier to collect or that is a superior fault predictor

D. Widely used metrics still have limitations

Software metrics exist as they are ought to enrich our abilities in managing various traits of software and its associated processes. Therefore, one of the crucial yardstick for evaluating their success is the extent of their usage across the software industry. Going by this principle as an evaluation mechanism accompanied by the experience and literature review, below are the key metrics that can be regarded as “the most successful metrics” in the world of software metrics:

- Lines of Code - LOC: in spite of the many arguments against it, this metric is still widely used and is the simplest one to gauge various software product and process attributes. It is easy and quick to measure, one of the easiest ones to comprehend even by novice practitioners. The metrics has seen varied flavors with evolving programming languages. Various supporting and contradicting views, but has strongly sustained.
- Metrics Related to Defect Counts: Software defects have been the closest associate for software products and have been leveraged as a basic measure to derive various quality related metrics. Whether it be quality of the in-process/final product or the quality of work delivered by programmers, defects serve as a key guiding parameter. Defect Removal Efficiency/Effectiveness (DRE) has been a crucial metric for assessing the quality of work being delivered by software teams throughout the software development life cycle.
- McCabe’s Cyclomatic Complexity: This metric thus far remains one of the most popular metrics in the area of static quality measurement of software products. In spite of the various criticisms, it saw, it has been popular and in wide usage due to its easier grasping and quicker ways of computing. On the limitations side, this metric indicates the degree of a program’s control complexity ignoring the data complexity part. It places same weight

on nested as well as non-nested loops. The metric may provide a misleading view with regards to a lot of simple comparisons and decision structures.

- **Function Points (FP counts):** FP analysis is quite complex and is hard to do properly when used for effort/resource estimation. Still in spite of these drawbacks, function points have been commonly recognized as a standard sizing measure.

E. Difference of perspective in outsourcing context

Measurements play a special role in keeping software projects under control especially in an outsourcing scenario. In such a scenario, where an organization has outsourced its software engineering activities either wholly or partially, will require vendor organization to be able to provide accurate project status to their customer organization in objective terms at any point in time. Similarly, customer organizations would always like to have quicker insights in to the status of outsourced activities, gauge the business value being delivered by vendor and assess the return on their investment towards outsourcing initiative. Yet you will come across many project managers managing such outsourced projects, who are still struggling to understand the basic essence of the purpose of metrics adoption in such projects, how to design environment specific metrics programs and making a selection of the most apt metrics set for their specific needs. One key cause for this project manager's jeopardy has been, unavailability of working guidelines and guidance model, which they can depend upon for making various decisions related to metrics selection and deployment.

As per Van Ekris [65], in an outsourcing context, customers will not only like to see a high quality end-product, they would like to have a continuous insight and confidence on the intermediate product quality throughout the software life cycle, how the in-process quality is being nurtured in to their products. Hence, in such scenarios, managers will need to identify ways on how their on-the-ground project execution aligns with overall theme of value optimization at customer organization. Projects/programs that are unable to deliver such an alignment will struggle in attaining customer confidence and will seldom result in higher levels of customer satisfaction.

The key expectations mentioned above in an outsourcing environment are challenged by some of the basic structural intricacies of such an environment. Firstly, the distance amongst organizations whether physical or virtual, vendor teams not being part of the teams at customer organization, frequently, spread geographically, causing a physical and often cultural distance. Surely, such kinds of distances pose newer challenges related to software development coordination: absence of ample informal interactions, informal relationships, which pose coordination and communication challenges and even the trust factor in outsourced engagements.

Control with metrics is a key component of success in outsourced software engineering programs. However, in spite of a realization of their importance and gamut of books and literature availability related to software metrics, managers and practitioners still face challenges in applying them in practice. The main reasons behind this can be attributed to primarily 2 grounds. First is that though a large body of metrics literature exists, there are so many metrics to confuse the practitioner community and hence, the selection remains sub-optimum. The next main reason is the outsourcing environment itself, where though the vendor teams attempt to bring in metrics based process improvement initiatives at vendor's side, the actual business value generation is expected to happen at customer's side.

F. Absence of a guiding model to predict quality

Software testing enjoys a crucial role and position in the overall software engineering umbrella. Software test managers are often challenged by their ability to do more and more with less and less. Their ability to predict end-product quality based on the analysis and trends of defects reported till now is considered as a distinguishing excellence parameter while qualifying a test manager. Metrics, whether related to effort/cost, testing hours, defects counts, tester efficiency and effectiveness often play a crucial role in equipping a test manager to enter the zone of excellence.

Several papers are published on the subject of measurement in software testing as a process. Yet, there are few concrete metrics and measurements to precisely and objectively assess the efficacy or value of software testing activity performed for a software product. Besides, there is no conformity, leave alone standards, concerning exactly what can and should be the focus of measurement to serve as a meaningful indicator that software has been "fully" tested or safe to use (or non-vulnerable). Project managers need to assess not just ROI (Return on Investment), but attributes like accuracy and meeting customer expectations. It would be beneficial to recognize how much testing is actually worth the risk of, say, \$5 million? We do not have an effective model of predictive software quality in place? Such a model is needed to guide test managers with situations like - For example if some specific scope of testing is executed, what is the probability of some particular level of quality? The industry today is really looking forward to have such a guiding models in place.

Quality Assurance (QA) plays a significant role during the entire software engineering process. Estimations shared by NIST (National Institute of Standards & Technology) signify that defective software results in yearly losses of one billion dollar plus only in USA. During the past decades, researchers have been focusing on

numerous themes within software testing and quality assurance. Some of the results shared have indicated a common goal amongst the development and quality assurance processes in software development organizations, whereas others do not. Testing and Quality assurance within the software development gain from well-established theoretical frameworks. Most of them are already integrated into practice. Nonetheless there are limitations of present day research direction and adoption.

Ultimately, software metrics research always had one of its primary objectives as to produce metrics that enable predicting quality of the software, while it is being developed (sometimes known as in-process quality). Defects detected in operation must always be differentiated from defects detected during the development process.

In the software engineering arena and especially in the area of software testing, the 'software metrics' approach has been a dominant approach to measurement. Over the decades, hundreds of 'software metrics' have been invented both by researchers and practitioners. Many of these metrics have been devised based either on researchers' institution or are having an empirical basis, or both. In their assessment of certain of these metrics, researchers have generally utilized the concepts of 'measurement theory' as the basis for their analysis. Though relevant, 'measurement theory' deals with just a subsection of the classical measurement concepts; researchers in the software metrics field, by focusing exclusively on 'measurement theory', have explored mainly the representation conditions, the mathematical attributes of crunching of numbers and suitable conditions for such manipulations

Likewise, even after thirty five years of data gathering and research, function point community is still grappling to come up with an enhanced framework with error ranges of lower than 25% for 75% of data in available samples (normally in engineering arena, error limits are anticipated to be much lower than 25%).

G. Inadequate support for decision making

Decision-making being a complex process, it is immature to assume that it can be based exclusively on structured data such as metrics. Yet, metrics play a useful role in decision making by software managers, i.e. metrics are an intrinsic component of the management information systems used by software managers. The test for all metrics researches however is to bring forward results of the research effort to the point where it is utilizable by software managers without them having to comprehend the underlying theory. Research and practice in the field of software metrics has helped in building up an empirical base for software engineering. This has been an important accomplishment [33]. However, the classical statistical approaches hardly provide managers with decision support required for risk evaluation and forecasting, and hence do not meet one of the most significant goals of software metrics. Perhaps the greatest challenge for medium to long term future of software metrics is to utilize them in such a way that they enable meet this key goal. An optimized software organization would always intend to keep a balance amongst several aspects of software product and process health e.g. quality, cost and schedule. Most of the metrics frameworks and models recommended in literature generally focus on sets of metrics, which help gauging various dimensions independently. There is need of a scientifically validated framework, which software practitioners can use to compare and take well-informed decisions after taking in to account various complementary metrics.

V. METRICS RESEARCH – DIRECTIONS FOR FUTURE

Gap identification based on the literature review carried out suggests that the subject area of software metrics has not yet been fully dealt with by current research, and much work is yet to be done to support software engineering as a scientific discipline on the basis of quantitative data and appropriate measurement methods satisfying the classic set of norms for measuring instruments as laid down by metrology body of knowledge in various engineering disciplines. There is a dire need to address the gaps, which continue to persist in the focus and approach by research and practice. There is a need to synthesize knowledge associated with the number of frameworks and recommendations invented by research community and recorded in literature. Such an amalgamation exercise needs to be done closely aligned with the way practitioner community makes use of metrics.

The amplified software metrics activity seen in industrial arena during last 15-20 years may be considered as a successful symptom of a growing subject area. Still this industrial practice is not correlated in content to the augmented academic and research initiatives. The 'metrics' in use today are essentially the same set of basic metrics that got introduced during the late 1960's. Most of the metrics in practice today still revolve around basic software product size measure viz. lines of code, number/types of defects in software products and estimated/actual efforts in terms of person hours or days.

Most of the academic research related to software metrics though focused on studying varied attributes of software and associated processes, but it failed to get a complete buy-in from industrial people. These are not encouraging happenings given that most of the metrics research outcome is actually projected to yield benefits when adopted in actual practice, otherwise, the enormous effort put in by academic research community goes for a waste. Seems like industrial community has given their acceptance to more basic, comprehensible metrics and refused to adopt metrics, which lacked clarity of purpose, are difficult to gather and are not deemed

completely valid. This phenomenon perhaps indicates that industrial people are being more practical regarding their choice of metrics. This shows that a key research goal has to devise metrics-based management decision support systems, which are based on simple yet meaningful metrics, with which industrial people can relate more empathetically.

In fact, the future of software metrics lies in using somewhat simpler existing metrics to build management decision-support systems that associate different facets of software development and testing and equip managers to make different kinds of evaluations, trade-offs and predictions during the software life-cycle.

There is necessity of a well-organized two-pronged approach, which should be targeted at both the scientific (research) and practicing (industry) communities, giving two main purposes as illustrated in Figure-2 below:

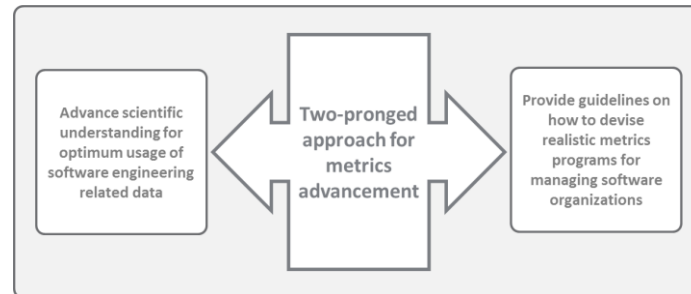


Figure-2. Two-pronged approach for metrics advancement

There is an urgent need to have a recommendation model in place that enables gain deeper insight into how a metrics program can be valuable in managing and cultivating software organizations further. For projects with outsourcing needs, such a model should enable vendor teams work in way that the software that they produce aligns closely with business goals of customer organization. Hence, there is a necessity for a framework or model founded on software metrics, which empowers managers for quicker human judgment based on metrics processing. The model should have a refreshed basis rather than outdated metrics approaches, which are generally based on regression-based models for effort/cost estimation and defect prediction, and yield meagre support for managers desiring to use measurement to assess and minimize risks. The model should take in to account the key influences of causation, ambiguity and associated subjective issues.

In [45], five major areas have been recognized, which are recommended as promising future research areas in the field of metrics related to software testing and quality assurance domains. Following is a brief description of these areas along with current gaps seen:

1. Static quality forecasting: In this area, research with occasionally promising results has been done, but still the practicing community has been rarely able to use these results in practice. One of the key reasons here has been that it is generally more cost effective to depend on developer verdict for such kind of predictions. Moreover, research could not crack threats of reliability and many a times metrics will not capture the core of whatever they are supposed to. As the field of static quality prediction remains an important one for practitioner community, metrics that capture this essence are very much needed. Analysis of process indicators might be the very first step in this direction.
2. Dynamic quality forecasting: This area has been considerably addressed by research community and is much in adoption by practitioners. Practitioners' requirements can be simply covered if prevailing theoretical and empirically validated frameworks and techniques are applied rightly. Current challenges for practitioners lie however, besides the right application, in data gathering and not being able to comprehend benefits of this approach.
3. Different types of test techniques: For commonly used test techniques, such as inspections, reviews, regression or user-acceptance testing, detailed evaluation of efficiency and effectiveness exists. Practitioners are already using the results and techniques quite commonly. Since these techniques are considered to be at quite a mature level, not much effort is seen for enhancing them further, apart from their continuous adoption to newer approaches viz. model-driven development.
4. Integrated decision models for test strategizing: There is need for integrated models, which help test managers select optimal test sets and test techniques resulting in optimum product quality with least effort/cost for testing. From a practitioner's viewpoint, the models available currently are very generic and, hence lack acceptance. Therefore, decisions that are made are many a times non-optimum. Nonetheless, this area is highly relevant to pursue as a future research area. For instance, it is well accepted that finding and fixing defects during earlier phases of software development life cycle saves money and ensures higher product quality. From practitioner's point of view, however, methods, which quantify these principles for decision making during test management are needed.
5. Integrated cost models for software quality: There already exist models, which deal with cost modelling for the software development process, such as CoCoMo-II and specifically COQUALMO for modelling quality

costs. But most of these approaches are too generic or cannot be adopted as such. Hence, there is a need of developing more versatile models with much closer association with intricacies of software engineering process as a whole.

The approach suggested in literature is to handle the key considerations largely overlooked in the usual metrics models namely: causality, ambiguity and different associated subjective issues. Therefore, the way ahead for research on software metrics lies in causal modelling, empirical software engineering, and multi-measure decision aids. In addition, practice needs to understand that theory requires data for developing models, which are best fit for practitioners' needs. At present, practice makes a complaint that useful models do not exist, but also does not realize the need for gathering data.

VI. CONCLUSIONS

In spite of a wealth of theoretical literature and research results available on software metrics, the field has not been able to come out of basic level of proficiency. Managers are still struggling to find that most "apt" set of software metrics for their needs. Situations become even more challenging in outsourced environments, which have their own set of environmental intricacies and customer expectations. Software industry has seen very slow growth in the software metrics inventions over past three decades and most of the metrics in adoption today are the same old metrics of 1970's with little enhancements. There are multiple definitions and interpretations of several metrics and many areas with no associated metrics defined at all. While the academia has seen a high volume of research effort, but the outcome of this research work has rarely been adopted even partially by the industrial community. There is a dire need of a close association between academia and industry on how to take the field of metrics forward so that the results obtained are aligned with the actual practical needs of practitioner community. In addition, there is need of a guiding model, which is based on simple basic measures/metrics and which can be used by the practitioner community as a directional aid while making decisions.

References

- [1] Abdel-Hamid, T. K. (1996). The slippery path to productivity improvement. *Software, IEEE*, 13(4), 43-52.
- [2] Abran, A., Sellami, A., & Suryn, W. (2003, September). Metrology, measurement and metrics in software engineering. In *Software Metrics Symposium, 2003. Proceedings. Ninth International* (pp. 2-11). IEEE.
- [3] Adams, E. N. (1984). Optimizing preventive service of software products. *IBM Journal of Research and Development*, 28(1), 2-14.
- [4] Akiyama, F. (1971, August). An Example of Software System Debugging. In *IFIP Congress (1)* (Vol. 71, pp. 353-359)
- [5] Bache, R., & Neil, M. (1995). Introducing metrics into industry: a perspective on GQM. *Software Quality Assurance and Measurement: A Worldwide Perspective*, 59-68.
- [6] Baker, A. L., Bieman, J. M., Fenton, N., Gustafson, D. A., Melton, A., & Whitty, R. (1990). A philosophy for software measurement. *Journal of Systems and Software*, 12(3), 277-281.
- [7] Basili, V. R., & Reiter Jr, R. W. (1981). A controlled experiment quantitatively comparing software development approaches. *Software Engineering, IEEE Transactions on*, (3), 299-320.
- [8] Basili, V. R., & Rombach, H. D. (1988). The TAME project: Towards improvement-oriented software environments. *Software Engineering, IEEE Transactions on*, 14(6), 758-773.
- [9] Bertolino, A. (2003, January). Software testing research and practice. In *Abstract State Machines 2003* (pp. 1-21). Springer Berlin Heidelberg.
- [10] Boehm, B. W. (1981). *Software engineering economics*.
- [11] Bouwers, E., Deursen, A. V., & Visser, J. (2013, May). Evaluating usefulness of software metrics: an industrial experience report. In *Proceedings of the 2013 International Conference on Software Engineering* (pp. 921-930). IEEE Press.
- [12] Briand, L. C., Morasca, S., & Basili, V. R. (1996). Property-based software engineering measurement. *Software Engineering, IEEE Transactions on*, 22(1), 68-86.
- [13] British Standard BS7925-1 (1998) *Software Testing vocabulary*.
- [14] Cavano, J. P., & McCall, J. A. (1978). A framework for the measurement of software quality. *ACM SIGSOFT Software Engineering Notes*, 3(5), 133-139.
- [15] Cem Kaner and Walter P. Bond. *Software engineering metrics: What do they measure and how do we know?* In *Proceedings of the 10th International Software Metrics Symposium*. IEEE CS Press, 2004
- [16] Coleman, D., Ash, D., Lowther, B., & Oman, P. (1994). Using metrics to evaluate software system maintainability. *Computer*, 27(8), 44-49.
- [17] Datta, S. (2009). Metrics and techniques to guide software development.
- [18] Dekkers, C. A. (1999). The secrets of highly successful measurement programs. *Cutter IT Journal*, 12, 29-35.
- [19] El-Emam, K. (2000). A methodology for validating software product metrics
- [20] Emanuel, A. W. R., Wardoyo, R., Istiyanto, J. E., & Mustofa, K. (2011). Statistical Analysis on Software Metrics Affecting Modularity in Open Source Software. *International Journal of Computer Science & Information Technology*, 3(3)
- [21] Fenton, N. (1994). Software measurement: A necessary scientific basis. *Software Engineering, IEEE Transactions on*, 20(3), 199-206.
- [22] Fenton, N. E., & Neil, M. (1999). A critique of software defect prediction models. *Software Engineering, IEEE Transactions on*, 25(5), 675-689.
- [23] Fenton, N. E., & Neil, M. (1999). Software metrics: successes, failures and new directions. *Journal of Systems and Software*, 47(2), 149-157.
- [24] Fenton, N. E., & Neil, M. (2000, May). Software metrics: roadmap. In *Proceedings of the Conference on the Future of Software Engineering* (pp. 357-370). ACM.
- [25] Fenton, N. E., & Ohlsson, N. (2000). Quantitative analysis of faults and failures in a complex software system. *Software Engineering, IEEE Transactions on*, 26(8), 797-814.
- [26] Fenton, N. E., & Pfleger, S. L. (1998). *Software metrics: a rigorous and practical approach*. PWS Publishing Co..

- [27] Frederiksen, H. D. USING METRICS IN MANAGING AND IMPROVING SOFTWARE DEVELOPMENT PROCESSES
- [28] G. Gordon Schulmeyer, James I. McManus, Handbook of Software Quality Assurance (1998), 3rd Edition, Prentice Hall PTR, Upper Saddle River, NJ.
- [29] Gilb, T., & Finzi, S. (1988). Principles of software engineering management (Vol. 4). Reading, MA: Addison-Wesley.
- [30] Gilb T, (1976) Software Metrics, Chartwell-Bratt
- [31] Gilb, T., & Graham, D. (1993). Software inspection (Vol. 34). S. Finzi (Ed.). Reading: Addison-Wesley
- [32] Glass RL, "A tabulation of topics where software practice leads software theory", J Systems Software, 25, 219-222, 1994.
- [33] Goodman, P. (1993). The Practical Implementation of Software Metrics. McGraw-Hill, Inc..
- [34] Grady, R. B. (1992). Practical software metrics for project management and process improvement. Prentice-Hall, Inc..
- [35] Hall, T., & Fenton, N. (1997). Implementing effective software metrics programs. IEEE software, 14(2), 55-65.
- [36] Hetzel, B. (1993). Making software measurement work: Building an effective measurement program. John Wiley & Sons, Inc..
- [37] Hoffman, D. (2000). The darker side of metrics. In Pacific Northwest Software Quality Conference, Portland, Oregon.
- [38] Horgan, J. R., London, S., & Lyu, M. R. (1994). Achieving software quality with testing coverage measures. Computer, 27(9), 60-69.
- [39] Horvitz, E., & Barry, M. (1995, August). Display of information for time-critical decision making. In Proceedings of the Eleventh conference on Uncertainty in artificial intelligence (pp. 296-305). Morgan Kaufmann Publishers Inc..
- [40] IEEE Standard Glossary of Software Engineering Terminology, ANSI/IEEE Std 610-1990, The Institute of Electrical and Electronics Engineers, New York, NY
- [41] ISO/IEC 15939:2002 (E), International Standard, Software Engineering – Software Measurement Process.
- [42] Jones, C. (1997). Strengths and weaknesses of software metrics. AMERICAN PROGRAMMER, 10, 44-49.
- [43] Jureczko, M. (2011). Significance of different software metrics in defect prediction. Software Engineering: An International Journal, 1(1), 86-95.
- [44] Kaner, C., & Bond, W. P. (2004). Software engineering metrics: What do they measure and how do we know?. methodology, 8, 6.
- [45] Karg, L., & Voelz, D. (2007). Scientific software test management-a research agenda. In Software Engineering (Workshops) (pp. 65-70)
- [46] Kitchenham, B., Pfleeger, S. L., & Fenton, N. (1995). Towards a framework for software measurement validation. Software Engineering, IEEE Transactions on, 21(12), 929-944
- [47] Lazic L., Mastorakis N., Cost Effective Software Test Metrics, WSEAS TRANSACTIONS on COMPUTERS, Issue 6, Volume 7, June 2008, pp. 599-619
- [48] Luo, L. (2001). Software testing techniques. Institute for software research international Carnegie mellon university Pittsburgh, PA, 15232(1-19), 19.
- [49] Mair, C., & Shepperd, M. (2011, May). Human judgement and software metrics: vision for the future. In Proceedings of the 2nd international workshop on emerging trends in software metrics (pp. 81-84). ACM.
- [50] McCabe, T. J., & Butler, C. W. (1989). Design complexity measurement and testing. Communications of the ACM, 32(12), 1415-1425.
- [51] McCabe, T. J. (1976). A complexity measure. Software Engineering, IEEE Transactions on, (4), 308-320.
- [52] Menzies, T., Di Stefano, J. S., Chapman, M., & McGill, K. (2002, December). Metrics that matter. In Software Engineering Workshop, 2002. Proceedings. 27th Annual NASA Goddard/IEEE (pp. 51-57). IEEE.
- [53] Munson, J. C., & Khoshgoftaar, T. M. (1992). The detection of fault-prone programs. Software Engineering, IEEE Transactions on, 18(5), 423-433.
- [54] Nagappan, N., Williams, L., Vouk, M., & Osborne, J. (2005, May). Early estimation of software quality using in-process testing metrics: a controlled case study. In ACM SIGSOFT Software Engineering Notes (Vol. 30, No. 4, pp. 1-7). ACM.
- [55] Ó Cinnéide, M., Tratt, L., Harman, M., Counsell, S., & Hemati Moghadam, I. (2012, September). Experimental assessment of software metrics using automated refactoring. In Proceedings of the ACM-IEEE international symposium on Empirical software engineering and measurement (pp. 49-58). ACM.
- [56] Ostrand, T., & Weyuker, E. (2010, November). Software testing research and software engineering education. In Proceedings of the FSE/SDP workshop on Future of software engineering research (pp. 273-276). ACM.
- [57] Ponisio, L., & van Eck, P. (2012). Metrics-based control in outsourced software development projects. IET software, 6(5), 438-450.
- [58] Putnam, L. H. (1978). A general empirical solution to the macro software sizing and estimating problem. IEEE transactions on Software Engineering, 4(4), 345-361.
- [59] Rawat, M. S., MGM's, C. O. E. T., Mittal, A., & Dubey, S. K. (2012). Survey on Impact of Software Metrics on Software Quality. IJACSA) International Journal of Advanced Computer Science and Applications, 3(1).
- [60] Rook, P. (1986). Controlling software projects. Software Engineering Journal, 1(1), 7.
- [61] Rosenberg Dr. Linda, Hammer Tad, Shaw Jak, (1999), Software Metrics and Reliability
- [62] Sedigh-Ali, S., Ghafoor, A., & Paul, R. A. (2001). Software engineering metrics for COTS-based systems. Computer, 34(5), 44-50.
- [63] Shahid, M., Ibrahim, S., & Mahrin, M. N. R. (2011). A study on test coverage in software testing. Advanced Informatics School (AIS), Universiti Teknologi Malaysia, International Campus, Jalan Semarak, Kuala Lumpur, Malaysia.
- [64] Staats, M., Whalen, M. W., Rajan, A., & Heimdahl, M. P. E. (2010, April). Coverage Metrics for Requirements-Based Testing: Evaluation of Effectiveness. In NASA Formal Methods (pp. 161-170).
- [65] Van Ekris, J. (2008, February). Customer perception of delivery quality: a necessary area for attention for project managers. In Proceedings of IASTED International Conference on Software Engineering as part of the 26th IASTED International Multi-Conference on Applied Informatics, Innsbruck, Austria (pp. 268-275)