Total Quality Management
Vol. 17, No. 5, 609–622, June 2006

Software Metrics for Enhanced Business Excellence: An Investigation of Research Issues from a Macro Perspective

G. S. SURESHCHANDAR & RAINER LEISTEN

Institut für Logistik und Informationsmanagement, Universität Duisburg-Essen, Germany

ABSTRACT *Software measurement is a challenging as well as an essential element of a wholesome and highly efficient software engineering culture. 'Software metrics' refers to the collective term underlying the wide range of activities associated with measurement in software engineering. There are numerous taxonomies to organize software metrics (based on entities, type of measure etc), but all of them fundamentally fit into three broad domains, namely resource, process and product metrics. The number of metrics in each of these categories is ever increasing with the result that practitioners are overwhelmed with data as they get quantitative inputs on a variety of attributes. This poses a real problem for decision-makers and analysts as many times it is unclear which attributes should be managed so as to augment the overarching goal of productivity or quality improvement. This paper aspires to address the research issues with respect to ably managing the subject of software metrics in a structured and coordinated manner. The present work deals with the concept of software measurement and metrics and argues that a great degree of simplification is required as, otherwise, the cornucopia of metrics would defeat the very purpose of the measurement approach, which is to quantify the key characteristics of interest in order to extract meaningful inferences about them and their relationships with one another. A cause-analytic model is also proposed, which if validated empirically would help to throw light on certain key metrics that could form the foundation for the further realization of the measurement and metrics approach. To encapsulate, the proposed research adopts a holistic approach in that it encompasses all categories of metrics in its analysis and it takes a macro-perspective in the sense that the sole purpose is to better organize the body of research with respect to software metrics.*

KEY WORDS: Software measurement, metrics, software quality, model, path analysis, total quality management, service quality

Introduction

In this era of intense competition, characterized by an atmosphere of globalization, liberalization and knowledgeable customers, corporations around the world vie hard towards

Correspondence Address: G. S. Sureshchandar, Fakultät für Betriebswirtschaft, Institut für Logistik und Informationsmanagement, Universität Duisburg-Essen, Lotharstr. 65, 47048, Duisburg, Germany. Email: suresh_gettala@hotmail.com

1478-3363 Print/1478-3371 Online/06/050609–14 © 2006 Taylor & Francis
DOI: 10.1080/14783360600588174

achieving greater honours in terms of business performance. As these corporations try to carve a niche for themselves in this race which has no finish line, ‘quality’ has probably become the hottest word in today’s corporate lexicon (Sureshchandar *et al.*, 2001c). The numerous approaches for managing quality effectively as propounded by various management gurus and researchers across the world (Crosby, 1979; Ishikawa, 1985; Deming, 1986; Taguchi, 1986; Juran *et al.*, 1988; and Feigenbaum, 1993) stand testimony to the significance of quality in both the business and academic milieu. Most of these techniques later blossomed into a philosophy called ‘Total Quality Management (TQM)’.

TQM is termed as an approach for continuously improving the quality of every aspect of the business life, i.e. it is a never-ending process of improvement for individuals, groups of people and the whole organization (Kanji & Asher, 1993, 1999). It is a holistic methodology towards achieving excellence in quality. It is multidisciplinary, organization-wide, people and process oriented (incorporating the human and technical aspects), customer-focused and emphasizes a cultural change that will result in all members of the organization striving all the time for continuous improvement (Ross, 1993). TQM, as a management concept, has got widespread accolades from practitioners and theoreticians alike.

The genesis of the movement dates back to the early 20th century when Walter Shewart, in the early 1920s, first introduced the concept of Statistical Process Control (SPC) to monitor quality in mass production manufacturing (Shewart, 1931). Many researchers have advocated various approaches, techniques, and organizational requirements for effective implementation of TQM in the manufacturing set-up. These approaches include top management commitment and leadership, quality policy, human resource management, product/service design, supplier quality management, process management, quality data and reporting, employee relations, workforce management, customer focus, customer involvement, benchmarking, statistical process control, employee satisfaction, employee empowerment, employee involvement, corporate quality culture, measurement (metrics) and analysis system, strategic quality management etc (Saraph *et al.*, 1989; Benson *et al.*, 1991; Flynn *et al.*, 1994; Powell, 1995; Ahire *et al.*, 1996; Black & Porter, 1996; Adam, 1994; Flynn *et al.*, 1994, 1995; Forker *et al.*, 1996; Madhu *et al.*, 1996; Samson & Terziovski, 1999; Terziovski & Samson, 1999; Agus *et al.*, 2000; Sureshchandar *et al.*, 2001c).

These dimensions are, in essence, tools of the intellect that were forged in the administrative theory, tempered in manufacturing quality management and therefore are naturally expected to be honed to cutting sharpness in service quality management. Per contra, although most of these dimensions and other techniques and strategies proposed by various theorists and practitioners seem to provide a near-universal remedy to the problems of the manufacturing business, keeping them as a complete yardstick for service quality improvement lacks sound logic (Sureshchandar *et al.*, 2001b). The reasoning, here is that although from a logical point of view most of the dimensions of manufacturing quality management should naturally apply to services, the transferability of manufacturing quality management dimensions to services calls for some serious soul-searching as services differ from manufacturing goods on a number of characteristics. Thus, the need and the logic for an independent treatment of services marketing centres on the subsistence of certain diverse attributes of services which are repeatedly cited in the literature: service intangibility, simultaneity of production, delivery and consumption, perishability, variability of expectations of the customers and the participatory role of customers in the service delivery (Sureshchandar *et al.*, 2001a, 2002).

The Software Industry Scenario

Among the service industries, the software industry perhaps poses the greatest challenge in terms of applying the TQM philosophy. This is more so because the software industry, in addition to possessing the above-mentioned unique features of service organizations, operates in a highly dynamic and volatile environment which in essence makes the implementation issues surrounding TQM much more complex than they actually are.

The IT industry is relatively naïve and immature, undergoing enormous and swift changes resulting in a profound effect on the attitude and behaviour of both the managers and the workforce. Due to this, managers, particularly in periods of economically glum times, have a tendency to think short term. Constant pressure to deliver is in continuous conflict with the need and rationale for standardization of methods/processes; statistically managing the design, development and the delivery of software; and the ability to learn from past experiences. Cultural change – which is the quintessence of TQM, therefore, is hard to be inculcated and arduous to be sustained (Koenigsberger, 1994). Furthermore, customers play the premier role in determining the quality (or lack of it) of the software. Conformance to customers' requirements and satisfying them to the utmost possible degree is probably the foremost thing that every software organization would aspire for. This is especially true in a software environment as most softwares are customized to cater to the requirements of one customer and if their needs are not addressed effectively, the system is bound to be a failure regardless of its technical capabilities (Caroll, 1995).

As a result, TQM as applied to the software industry is still very much in its infancy, although the applicability of TQM to software development has been advocated by many practitioners and quality consultants. Moreover, customers have started chanting the mantra of 'no certification no invitation to tender' which, in essence, lead to organizations striving hard on getting the ISO and CMM certifications (Koenigsberger, 1994). Thus, much of what we know about the use of quality management principles for software quality improvement has been limited to the establishment of processes and other documentation procedures that go in tune with the ISO and CMM requirements instead of the integrated TQM paradigm.

From a software industry perspective, the numerous TQM approaches that have been identified by various researchers could be grouped under five broad dimensions, namely, *Customer Focus, Soft issues* (*human aspects*) *of quality*, *Hard issues* (*process* – including both the business process and the product/service design, development and delivery process; and *servicescapes* – the tangible facets of the service environment), *Continuous Improvement* and *Measurement and Analysis*. These dimensions are portrayed in Figure 1. The importance of these dimensions has been thoroughly and exhaustively stated in the quality management literature.

The *soft issues* include aspects such as Top Management Commitment and Leadership, Human Resource Management Organizational Culture, Social Responsibility, etc, that will fall under the domain of 'Human Factors' of effective quality management. *Hard issues* basically refer to the process – comprising both the business process and the product/service design, development and delivery process. The key to TQM relies on using the process as a means to transfer knowledge thereby responding to the customers faster than the competitors. *Customer focus* should be an important sub-system of any organizational system supporting TQM because organizations can outscore their competitors by effectively addressing customers' needs with new ideas and technologies and

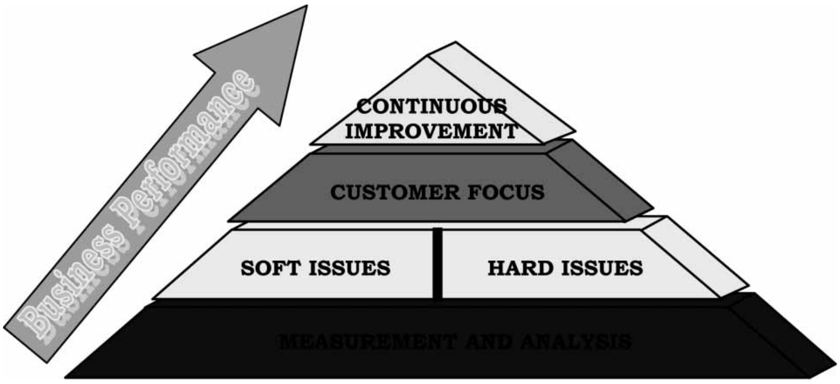

Figure 1. A holistic model for TQM in the software industry

produce products that placate or surpass customer expectations, and anticipate and react to customers' evolving interest and wants. TQM can only be effective if all the dimensions operate in an environment that embraces *continuous improvement* as a philosophy of quality management, i.e. the success of any TQM movement would largely depend on how synergically the various dimensions are espoused in an ambience of continuous improvement.

The soft issues, hard issues and all customer-related information should be quantified, measured and investigated in order to monitor, control and improve the characteristics of concern. Therefore, *Measurement and Analysis System* forms the cornerstone upon which all other dimensions are built. This can be attributed to the fact that most software processes and procedures are driven by approaches that could be effectively managed using quantitative or statistical techniques. For instance, all of the project, process and resource-oriented data have to be quantified so that one could investigate the progress of a particular project. In order to do so, the first prerequisite would be to define certain reliable and valid measures that will depict the parameters of interest.

Review of Literature on Measurement and Metrics

Software measurement is a challenging but an essential element of a wholesome and highly efficient software engineering culture. The fact that measurement is extremely vital to the progress of all sciences is an acknowledged statement (Kan, 1995). Kan (1995) further asserts that scientific development or progress is made only through observations and generalizations based on measurements and data, the derivation of theories as a result and in turn the confirmation or refutation of theories via hypothesis testing based on further empirical data. Ideally, measurement relates to the concept or entity of interest and the operational definition of the same. Depending on the operational definition, different scales (nominal, ordinal, interval and ratio) are used to portray various levels of measurement. The two most important criteria of assessing the quality of measurement are reliability and validity. *Reliability* refers to the consistency of the metric and the measurement method over repeated measurements while *validity* refers to the degree to which a measure measures what it is intended to measure. There are several ways by which reliability and validity can be established (Sureshchandar *et al.*, 2001c).

As is well known, software projects are notorious for running over budget and behind schedule. Software measurement helps to quantify the schedule, effort, size, defects and other measures of quality performance, thereby helping to have a better control over them. Software metrics refers to the collective term underlying the wide range of activities associated with measurement in software engineering. Although there are several nomenclatures to classify software metrics (for example based on entities, type of measure etc) all of them, in essence, would fit into three broad domains, namely *resource*, *process* and *product* metrics (Kan, 1995; Fenton & Pfleeger, 1996; Burr & Owen, 1996; and Pressman, 2001). Fenton & Pfleeger (1996) define products as any artefacts, deliverables or documents that result from a process, while processes are collections of software-related activities. Resources are entities required by a process activity. Use of metrics helps in goal setting, project planning and improving quality, productivity and customer satisfaction (Moller & Paulish, 1993).

Research, both theoretical and empirical, in software engineering and especially in software metrics, is indeed quite extensive, from the view point of investigating the different categories of metrics. There seems to be a considerable body of work analysing, products, processes and resources (Fenton & Pfleeger, 1996). Several works have dealt, in detail, with the *different variables* among the product, process and resource metrics categories. They have considered the effects of these variables on one another from a micro-perspective. For instance, some of the studies that focused on products analysed various approaches to improve software structure and maintenance (e.g. Henry *et al*., 1994). Many other studies have aspired to investigate prediction of fault density. Typical examples include a regression-based model predicting the number of faults (Akiyima, 1971), the relationship between fault density and size of software (Lipow, 1982), the effects of modification and reuse on fault density (Moller & Paulish, 1993; Hatton & Roberts, 1994). Studies on resource metrics analysed factors such as how personnel resources expend their time (Perry *et al*., 1994), ways of making effective inspections (Weller, 1993), etc. Similarly, many other efforts dealt with processes as well thanks to the heightened interest generated by the process improvement communities and the increased recognition of process validation certifications such as ISO 9000, CMM, CMMI etc, by both business and academia. These include studies like benefits of the object-oriented approach to build software (Stark, 1993), effects of inspections on code quality (Kitchenham & McDermid, 1986), and a classic clean-room approach that studied the influence of processes on fault densities (Linger, 1994).

Apart from studies on investigating the product, process and resource metrics, the software measurement literature also comprises other research works that explored both the *internal and external attributes* belonging to the above mentioned three groups of metrics. Although there are many ways by which one could explain the meaning of internal and external attributes, a classical definition of them was propounded by Fenton & Pfleeger (1996). The authors define internal attributes as 'those that can be measured purely by means of a product, process or resource itself'. Consequently, an internal attribute can be measured by examining the product, process or resource on its own, separate from its behaviour. External attributes, on the other hand, are 'those that can be measured only with respect to how the product, process, or resource relates to its environment'. Here, the behaviour of the process, product or resource is important rather than the entity itself.

Most such studies on attributes actually strived to analyse the impact of the attributes (internal or external) on software quality. For instance, McCall *et al*. (1977) proposed a set of factors, such as correctness, reliability, usability, integrity, efficiency, etc, among others, that affect software quality. The authors also came up with weights for the different metric based on local products and concerns. Another of the earlier works on the subject was carried out by Grady & Caswell (1987), who gave an acronym FURPS (i.e. functionality, usability, reliability, performance and supportability) that can be used to establish quality metrics for each step in the software engineering process. Some other factors have also been identified and used as part of the ISO standards (see ISO 9126). Whilst the above studies explored software metrics from a subjective perspective, there were a few other studies that detailed the importance and subsequent identification of more quantitative measures to depict the quality of softwares. Such works explicated the reason behind defining and using objective measures such as LOCs, Function Points, the bang metric etc (Cavano & McCall, 1978; DeMarco, 1982; Basili & Weiss, 1984; Ejiogu, 1991; and Roche, 1994).

There were also other works that dealt with some specific attributes, namely, software testing metrics (Pavur *et al*., 1999), software volatility (Zhang & Windsor, 2003), optimizing software quality (Babu & Nalina, 1996), and software benchmarking using Function Point analysis (Cheung *et al*., 1999) etc.

From the above discussions it is palpable that the literature on software metrics has indeed evolved over a period of years from the standpoint of churning out new metrics so as to define variables of interest in software engineering. Although the utility of such studies can never be disputed as they attempted to investigate current practices by evaluating real data in real situations, most of the metrics that have been identified could not provide practical support to the software engineer as some of them demanded measurement that is too complex, while others were so esoteric that few practising professionals were able to make real sense of them (Pressman, 2001).

To encapsulate, not withstanding the huge explosion of the metrics in recent years, the body of literature on software measurement and metrics is still devoid of an all encompassing study that would throw up a holistic model depicting the key/critical metrics in each of the three categories and their relationships with one another. Such a model, from a macro-perspective, would help to organize better the domain of literature in a more structured manner, thereby unearthing significant relationships among the various domains of metrics.

State-of-the-Art Practices with Respect to Software Metrics and the Research Problem

As already stated, with the software industry undergoing a metamorphosis at a rapid pace, identification and investigation of research issues with respect to software quality management assumes paramount prominence. Such efforts would also help practitioners to manage their software processes effectively. Of the critical issues one has to encounter in managing software quality, '*measurement and the analysis and interpretation of metrics data*' warrants great attention, as precise quantification of software quality seems to be an uphill task due to the highly abstract nature of the same.

The ensuing sections address this particular aspect of software quality management, which deals with problems associated with measurement and an effort has been made here to delve on the crucial role of software metrics and its subsequent analysis in improving software quality.

Software metrics as a subject is made up of two distinct phases.

1. The first deals with *defining specific measures* that are required to quantify the characteristics of interest.
2. The second deals with the *establishment of relationships* among the various categories (or domains) of metrics.

The challenges the metrics approach face today are with respect to both.

Challenges with Respect to the First Phase

The first phase of the metrics approach is reasonably well developed, based on a robust theoretical background, resulting in a huge number of metrics spanning the entire resource, product and process categories (see Table 1). It could be seen from the table that the number of metrics in each of the subcategories with respect to three domains is

Table 1. Classification of software measurement activities (adapted from Fenton & Pfleeger, 1996)

Entities	Attributes	
1. Products	Internal	External
Specifications	size, reuse, modularity, redundancy, functionality, syntactic correctness, . . .	comprehensibility, maintainability, . . .
Designs	size, reuse, modularity, coupling, cohesiveness, inheritance, functionality, . . .	quality, complexity, maintainability, . . .
Code	size, reuse, modularity, coupling, functionality, algorithmic complexity, control-flow structuredness, . . .	reliability, usability, maintainability,reusability
Test data	size, coverage level, . . .	quality, reusability, . . .
. . ..	. . ..	. . ..
2. Processes		
Constructing specification	time, effort, number of requirements changes,. . .	quality, cost, stability, . . .
Detailed design	time, effort, number of specification faults found, . . .	cost, cost-effectiveness, . . .
Testing	time, effort, number of coding faults found, . . .	cost, cost-effectiveness, stability, . . .
. . ..	. . ..	. . ..
3. Resources		
Personnel	age, price, . . .	productivity, experience, intelligence, . . .
Teams	size, communication level, structuredness, . . .	productivity, quality, . . .
Organizations	size, ISO Certification,CMM level	maturity, profitability, . . .
Software	price, size, . . .	usability, reliability, . . .
Hardware	price, speed, memory size, . . .	reliability, . . .
Offices	size, temperature, light, . . .	comfort, quality, . . .
. . ..	. . ..	. . ..

so large that a great degree of *simplification* is required as, otherwise, the cornucopia of metrics would defeat the very purpose of the measurement approach, which is to quantify the *key* characteristics of interest in order to extract meaningful inferences about them and their relationships with one another. Although it could be argued that the number and type of metrics used in practice with respect to each of the domains would depend on the objectives of the metrics programme, some degree of standardization/integration of the huge number of metrics is required. Typically, this would abet in improving the effectiveness of the metrics approach, as it would aid in the identification of the critical metrics within each domain that have to be targeted instead of harbouring a huge number of metrics.

For instance, studies show that there are several instances wherein developers/metrics implementers have expressed difficulties with measurement because they sometimes feel overwhelmed with data as they get quantitative feedback on a variety of attributes, and many times it is hazy as to which attributes should be dealt with in order to enhance quality or business performance. Hence, there have been many prescriptions to combine measures, into a single summary measure (Fenton & Pfleeger, 1996). This observation was also made by other authors (see Moller & Paulish, 1993) wherein they have emphasized the need for the use of a limited number of metrics.

Therefore, there is an immediate need to develop a simplistic approach to address the complex problem of dealing with the plethora of metrics that are available. Such an approach actually takes a *macro-viewpoint* as it stresses focusing on the vital few rather than on the trivial many in order to constitute a systematic and an effective measurement and analysis approach. Of course, several metrics pertaining to the specific problem of concern have to be developed in order to address the same, but at the same time there is also a need to identify some key fundamental metrics that are required for executing an effective metrics programme. A simplified set of metrics in each of the domains (Resource, Process and Product) would also provide a more definite basis for subsequent analyses. The metrics identified could span both the *internal* and *external attributes* categories.

Challenges with Respect to the Second Phase

Once the critical metrics are identified, the process of the measurement is no way complete. The second challenge is to establish the relationships among the various domains of metrics so that they can be effectively managed, which ultimately would result in quality improvement. This is highly imperative for the process of theory building. In addition, such an investigation would help to change the practitioner's outlook towards the metrics approach, as at present there seems to be an inherent tendency on the part of software practitioners to resist any metrics programme. Several reasons could be attributed for such a scenario. As the software process is a highly technical activity, the practitioners are more concerned about the technicalities of the software, with apprehensions that the metrics approach would aim on getting only the numbers right at the cost of producing good software. Secondly, the software industry is characterized by a high-pressure environment, overtime working, and intense slogging at the last minute in order to adhere to the strict delivery schedules. This hinders the precise and systematic collection of data, thereby resulting in the scarcity of data related to the software process. Statistical techniques, in order to draw inferences more accurately about the population parameters based on sample statistics, require huge volumes of data. Thirdly, most software projects are unique in one way or the other, making it difficult to compare data from two different projects.

Despite the inadequacies and the perceived lackadaisical approach of the practitioners towards the measurement approach, the software metrics body of knowledge has been swelling in recent years. Fenton & Pfleeger (1996) report evidence of several studies that investigated the relationships between different metrics within each of the domains. However, none of the reported studies explored the relationships among the various domains. Therefore, the research literature is not comprehensive as it fails to establish the links among the various categories of metrics in the first place (by taking a macro-viewpoint), before investigating the relationships among metrics within each of the categories. In addition, because of the fact that the total body of metrics is highly complex (as discussed in the preceding section), whatever investigations have been carried out so far have had little significance, both from the perspectives of the theoreticians and the practitioners. Therefore it would be more prudent to investigate the influences of the different domains of metrics on one another, after the simplification process (as in the preceding section) is carried out.

The relationships among the metrics seem to be pretty straightforward, although the nature and degree of relationships warrant meticulous scrutiny. Of the three domains, the *product* metrics gives valuable information on the various aspects of the product, thereby providing opportunities for improving product quality. The effectiveness of the structured *processes* in improving software product quality is well known (Humphrey, 1989; Fenton & Pfleeger, 1996; Burr & Owen, 1996; and Pressman, 2001). The link between the ISO/CMM certifications and achieving higher levels of software quality has been recorded in several research works (see Parzinger & Nath, 2000). A more fundamental question that needs to be deliberated at this juncture is irrespective of the ISO/CMM levels; does the process itself positively influence high levels of software quality? One possible link that could be investigated is the simple independent–dependent chain connecting process metrics and the product metrics. Process metrics enable an organization to take a strategic view by providing insights into the effectiveness of the software process. *Resources* are required to accomplish any activity. Therefore, resource metrics would influence both the above-discussed metrics' domains (product and process) independently.

In addition, each of the domains is made up of metrics that would measure both the internal and the external attributes. The literature on metrics stresses the need for measuring internal attribute measurements to support measurement and decision making about external attributes (Fenton & Pfleeger, 1996). The authors further state that one of the cardinal goals of software-measurement research is to establish the linkages among internal and external attributes, apart from identifying new and useful methods for directly measuring the attributes of interest.

Therefore, in line with the above findings a model (Figure 2) is proposed that depicts the relationships among the three domains of metrics. This model has to be validated and investigated in order that the simultaneous relationships among the metrics are brought to light. From a theoretical perspective, given the relative newness of the subject (in terms of better organization), this provides a basis for a further understanding of the utility of the different metrics for quality improvement. On the practitioner's front, as mentioned earlier, the laid-back attitude of the practitioners towards the metrics approach

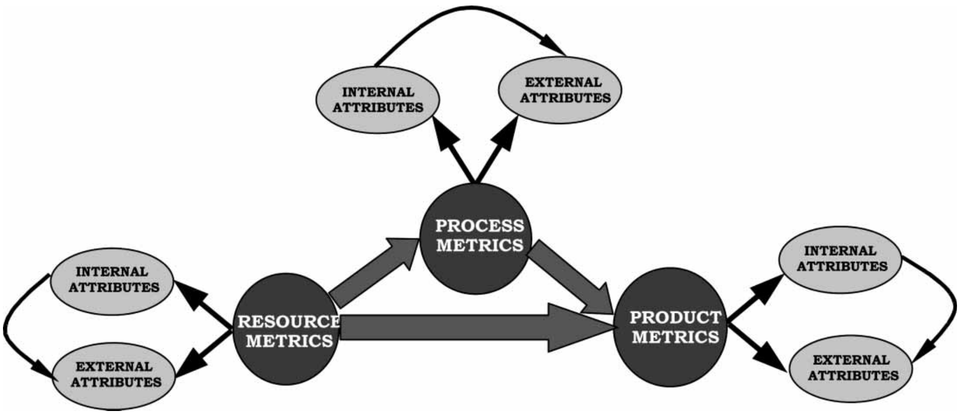

Figure 2. A causal model depicting the relationships among the three domains of metrics

could be because of the fear that the collected metrics may be used against them, and also it may take too much time to collect and analyse metrics data. The bottom line is that they are not fully aware of the usefulness of the metrics approach. A macro-level analysis such as the proposed research would help to allay most of these fears, vis-à-vis a micro-level investigation, which would only add more confusion, resulting in the nonchalance of the already less convinced practitioners.

Objectives and Methodology of the Proposed Research Work

Objectives

Based on the above discussions, the objectives of the research work are formulated in a two-fold manner.

1. To systematize the plethora of software metrics into a manageable set (unearth the critical summary measures) by taking a macro-viewpoint;
2. To make a detailed investigation of the relationships among the three domains of software metrics.

Methodology

As explained in the preceding section, the model that is proposed has to be empirically validated so as to be used effectively in any metrics approach. A crucial aspect in the evolution of a fundamental body of knowledge in any management theory is the development of genuine measures to obtain valid and reliable estimates of the constructs of interest and their relationships to one another (Sureshchandar *et al.*, 2001c). Therefore, research should initially identify the intrinsic dimensions (key metrics) with respect to the three domains, check that they are measured reliably and validly, and subsequently ascertain their influence on improving software product quality. The proposed model illustrates causal relationships among the metrics and hence 'Path Analysis' is planned to be used. The Path Analysis approach is confirmatory in nature and is also highly pertinent to the chosen problem as the purpose of the proposed study is to structure the huge body of metrics into a manageable set and not to unearth new metrics.

The data for the analysis is proposed to be collected from medium and large-scale software organizations. Historic data on metrics will be used and the data collected from the software companies would be compared with the same to better arrange the available body of knowledge. The possibility of trans-national studies could also be explored, which would investigate the data collected from two or more countries. Such trans-national studies would help to throw light on the various quantitative approaches to software quality management from a global perspective.

The Role of Statistics in Investigating Metrics Data

Although the number of metrics that have been identified so far is quite voluminous from an academic perspective, the collection, analysis and management of metrics in practice have left much to be desired. The software practitioners appear to have a laid-back propensity towards any metrics programme. This may be because of the fear that the collected metrics may be used against them, and also it may take too much time to collect and analyse metrics data. The bottom line is that they are not fully aware of the usefulness of the metrics approach. Strangely enough, the use of statistics (which is a powerful science having a wide range of applications in almost all business and economic problems) has been highly limited with respect to analysing quality data in the software industry. As we all know, statistical techniques refer to the wide gamut of techniques that help to investigate simple and complex business problems quantitatively, based on a statistical basis.

The applicability of statistical techniques in analysing metrics data has been largely confined to the seven basic tools of quality (namely, Check sheets, Scatter diagrams, Histograms, Pareto diagrams, Fish-bone diagrams, Run charts and Control charts). Control charts, by far, remain the most widely used tool for analysing data for software process improvement (Burr & Owen, 1996). Both types of control charts, namely control charts for variables and control charts for attributes, are used to check for the stability and the capability of software processes. There is no doubt that a control chart is a powerful tool in its own right, to monitor and control process behaviour, but it is rather surprising to note that other advanced statistical techniques are seldom used in managing software quality, despite the fact that the efficacy of such techniques in providing great insights on the problem of concern is well known.

Although, there are instances where techniques like Linear Regression have been used in defect prediction (e.g. defects $= f(\text{size})$) and effort prediction models (e.g. effort $= f(\text{size})$), it is a verity that the software industry has not fully utilized the cornucopia of statistical techniques that are available. Even univariate techniques like 't' tests and one-way ANOVA have been rarely made use of in analysing metrics data. These techniques could help to throw light on so many parameters of significance. For instance, one of the main priorities of software quality improvement may be to identify those phases of the life cycle that need immediate attention with respect to some established targets. To elaborate further, one may be interested in identifying that particular phase of the life cycle that contributes more to the defects. This can be found by means of, say, Pareto analysis, but techniques like ANOVA and the post-hoc tests such as Bonferroni and Duncan may be used to ensure that they are done statistically.

In addition, techniques such as multiple regression can be used to establish so many other relationships apart from just defect and effort predictions. For example, one may be interested studying the multiple simultaneous influences of size, effort, defects, software complexity etc on, say, software quality or productivity. Complex causal relationships involving more than one independent-dependent variable chain can be investigated by means of techniques such as path analysis.

Consequently, there seems to be a great potential in these statistical techniques to provide a statistical approach for investigating the problems with respect to managing software quality in a structured manner. Therefore, research should strive to unravel the links between the various metrics (both basic and derived) through the use of advanced statistical techniques. Such statistical models could aid providing a quantitative basis for investigating and understanding the complex relationships among the different metrics.

Summary

Software companies, of late, are just beginning to get to grips with the truth behind what their manufacturing counterparts learned in the past few decades – quality doesn't improve unless it is measured and managed. However, although software metrics as a subject area is more than three decades old, it has hardly diffused into mainstream software quality management in practice. This may be because most software metrics methodology has failed to address the fundamental aspect of its purpose, namely to collect adequate and pertinent data in order to support quantitative/statistical decision analysis throughout the entire duration of the software life cycle.

Even the recent upsurge in software metrics activity may be just due to the need to do something different when things are in bad shape or to satisfy the requirements of quality accreditations like the ISO, CMM, CMMI and the like. This is typically exemplified by information from the US, which illustrated that the CMM has been the major driver of the metrics activity, because of the perceived positive relationship between the use of metrics and achieving higher levels of CMM (Humphrey, 1989). At the same time, these certifications do not explicitly call for the use of advanced statistical techniques, with the effect there has been a very limited use of complex statistical techniques for analysing metrics data. To summarize, although the aforementioned causes have hampered the effective adoption of the metrics methodology in practice, they are in no way insurmountable. It only requires a dedicated effort and the collective wisdom of the software practitioners so that a proper 'Measurement and Analysis System' is put in place. In order to do that the entire software populace will have to be first convinced about the efficacy of such an approach. Academic research on the utility of advanced statistical models in the analysis of metrics data would go a long way in ensuring the same.

This paper has presented a framework that addresses the research issues with respect to ably managing the subject of software metrics in a structured and coordinated manner. The results of the proposed study would help to effectively investigate the vast body of knowledge with respect to software metrics in the form of a unified simplistic model. In this vein, it would help to throw light on certain key metrics, which would form the foundation for the further realization of the measurement and metrics approach. The purpose is to come up with a model that could provide a statistical basis for providing a universal solution to the problems of quantitative decision-making in effectively managing software quality.

It would also help software practitioners to better comprehend the logic and rationale behind a systematic and well-structured metrics approach so that it results in the periodic and precise collection of relevant data, which could be used to draw meaningful inferences about the targeted parameters. It would thus provide additional insights on the underlying concepts of the metrics approach as applied in the management of software quality. From a business perspective, these insights will provide crucial inputs on the financial, personnel, training and infrastructure requirements that are fundamental for success in the ever-changing software environment. Such inputs are extremely vital to brazen out the challenges posed by the emerging information technologies that continuously restructure the software landscape of the corporate world.

By adopting a macro-perspective, the proposed study strives to offer a comprehensive, and yet simple methodology for scientifically examining how the plethora of metrics can be structured into a systematic framework for the development of a conceptual and empirical understanding of the same.

References

Adam, E. E. Jr. (1994) Alternative quality improvement practices and organization performance, *Journal of Operations Management*, 12, pp. 27–44.

Agus, A. *et al.* (2000) The structural impact of total quality management on financial performance relative to competitors through customer satisfaction: a study of Malaysian manufacturing companies, *Total Quality Management*, 11(4/5&6), pp. S808–S819.

Ahire, L. S. *et al.* (1996) Development and validation of TQM implementation constructs, *Decision Sciences*, 27, pp. 23–56.

Akiyama, F. (1971) An example of software system debugging, *Information Processing*, 71, pp. 353–379.

Babu, A. J. G. & Nalina, S. (1996) Modelling and optimizing software quality, *International Journal of Quality and Reliability Management*, 13(3), pp. 95–103.

Basili, V. R. & Weiss, D. M. (1984) A methodology for collecting valid software engineering data, *IEEE Transactions Software Engineering*, 10, pp. 728–738.

Benson, G. P. *et al.* (1991) The effects of organizational context on quality management: an empirical investigation, *Management Science*, 37, pp. 1107–1124.

Black, A. S. & Porter, L. J. (1996) Identification of the critical factors of TQM, *Decision Sciences*, 27, pp. 1–22.

Burr, A. & Owen, M. (1996) *Statistical Methods for Software Quality – Using metrics for process improvement* (UK: International Thompson Computer Press).

Caroll, J. (1995) The application of total quality management to software development, *Information Technology and People*, 8(4), pp. 35–47.

Cavano, J. P. & McCall, J. A. (1978) A framework for the measurement of software quality, *Proceedings of the ACM Software Quality Assurance Workshop*, pp. 133–139.

Cheung, Y. *et al.* (1999) Software benchmarks using function point analysis, *Benchmarking: An International Journal*, 6(3), pp. 269–276.

Crosby, B. P. (1979) *Quality is Free* (New York: McGraw-Hill).

DeMarco, T. (1982) *Controlling Software Projects* (Yourdon Press).

Deming, E. W. (1986) *Out of the Crisis* (Cambridge, MA: MIT Centre for advanced Engineering).

Ejiogu, L. (1991) *Software Engineering with Formal Metrics* (QED Publishing).

Feigenbaum, A. V. (1993) *Total Quality Control*, 3rd edn (New York: McGraw-Hill).

Fenton, N. E. & Pfleeger, S. L. (1996) *Software Metrics – A Rigourous and Practical Approach*, 2nd edn (UK: International Thompson Computer Press).

Flynn, B. B. *et al.* (1994) A framework for quality management research and an associated measurement instrument, *Journal of Operations Management*, 11, pp. 339–366.

Flynn, B. B. *et al.* (1995) The impact of quality management practices on performance and competitive advantage, *Decision Sciences*, 26, pp. 659–691.

Forker, L. B. *et al.* (1996) The contribution of quality to business performance, *International Journal of Operations and Production Management*, 16, pp. 44–62.

Grady, R. B. & Caswell, D. L. (1987) *Software Metrics: Establishing a company-wide Program* (Prentice-Hall).

Hatton, L. & Roberts, A. (1994) How accurate is scientific software?, *IEEE Transactions on Software Engineering*, 20(10), pp. 785–797.

Henry, J. *et al.* (1994) Improving software maintenance at Martin Marietta, *IEEE Software*, 11(4), pp. 67–75.

Humphrey, W. S. (1989) *Managing the Software Process* (Reading, MA: Addison Wesley).
Ishikawa, K. (1985) *What is Total Quality Control? The Japanese Way* (Englewood Cliffs, NJ: Prentice Hall).
Juran, J. M. *et al.* (1988) *Juran's Quality Control Handbook*, 4th edn (New York: McGraw-Hill).
Kan, S. H. (1995) *Metrics and Models in Software Quality Engineering* (Boston, MA: Addison-Wesley).
Kanji, G. K. & Asher, M. (1993) *Total Quality Management Process: A Systematic Approach* (Oxford: Carfax).
Kanji, G. K. & Asher, M. (1999) *100 Methods for Total Quality Management* (New Delhi: Sage).
Kitchenham, B. A. & McDermid, J. A. (1986) Software metrics and integrated project support environments, *Software Engineering Journal*, 1(1), pp. 58–64.
Koenigsberger, J. G. (1994) Quality in a multinational IT services company – hard earned experiences. *Training for Quality*, 2(2), pp. 36–40.
Linger, R. C. (1994) Cleanroon process model, *IEEE Software*, 11(2).
Lipow, M. (1982) Number of faults per line of code, *IEEE Transactions on Software Engineering*, 8(4), pp. 437–439.
Madhu, C. N. *et al.* (1996) An empirical assessment of quality dimensions on organizational performance, *International Journal of Production Research*, 34, pp. 1943–1962.
McCall, J. *et al.* (1977) *Factors in Software Quality*, NTIS AD–A049–014, 015, 055, November.
Moller, K. H. & Paulish, D. J. (1993) *Software Metrics – A Practitioner's Guide to Improved Product Development* (UK: Chapman and Hall Computing).
Parzinger, M. J. & Nath (2000) A study of the relationships between total quality management implementation factors and software quality, *Total Quality Management*, 11(3), pp. 353–371.
Pavur, R. *et al.* (1999) Software testing metrics: do they have merit?, *Industrial Management and Data Systems*, 99(1), pp. 5–10.
Perry, D. E. *et al.* (1994) People, organizations and process improvement, *IEEE Software*, 11(4), pp. 36–45.
Powell, C. T. (1995) Total quality management as competitive advantage: a review and empirical study, *Strategic Management Journal*, 16, pp. 15–37.
Pressman, R. S. (2001) *Software Engineering – A Practitioner's Approach*, 5th edn (USA: McGraw Hill International Edition).
Roche, J. M. (1994) Software metrics and measurement principles, *Software Engineering Notes*, ACM, 19(1), pp. 76–85.
Ross, J. (1993) *Total Quality Management: Text, Cases & Reading* (Florida: St. Lucie Press).
Samson, D. & Terziovski, M. (1999) The relationship between total quality management practices and operational performance, *Journal of Operations Management*, 17, pp. 393–409.
Saraph, J. V. *et al.* (1989) An instrument for measuring the critical factors of quality management, *Decision Sciences*, 20, pp. 810–829.
Shewart, W. A. (1931) *Economic Control of Quality of Manufactured Product* (New York: Van Nostrand).
Stark, M (1993) Impact of object-oriented technologies: seven years of software engineering, *Journal of Systems and Software*, November.
Sureshchandar, G. S. *et al.* (2001a) Customer perceptions of service quality – a critique, *Total Quality Management*, 12(1), pp. 111–124.
Sureshchandar, G. S. *et al.* (2001b) A conceptual model for TQM in service organizations, *Total Quality Management*, 12(3), pp. 343–363.
Sureshchandar, G. S. *et al.* (2001c) A holistic model for total quality service, *International Journal of Service Industry Management*, 12(4), pp. 378–412.
Sureshchandar, G. S. *et al.* (2002) Determinants of customer perceived service quality: a confirmatory factor analysis approach, *Journal of Services Marketing*, 16(1), pp. 9–34.
Taguchi, G. (1986) *Introduction to Quality Engineering, Designing Quality into Products and Processes* (New York: Unipub).
Terziovski, M. & Samson, D. (1999) The link between total quality management practice and organizational performance, *Internal Journal of Quality and Reliability Management*, 16, pp. 226–237.
Weller, E. F. (1993) Lessons from three years of inspection data, *IEEE Software*, 10(5), pp. 38–45.
Zhang, X. & Windsor, J. (2003) An empirical analysis of software volatility and related factors, *Industrial Management & Data Systems*, 103(4), pp. 275–281.