Software Measurement Activities in Small and Medium Enterprises: an Empirical Assessment

O. Tolga Pusatli

Department of Mathematics and Computer Science Çankaya University Ankara Turkey, pusatli@cankaya.edu.tr

Sanjay Misra

Department of Computer Engineering Atilim University Ankara Turkey, smisra@atilim.edu.tr

Abstract: An empirical study for evaluating the proper implementation of measurement/metric programs in software companies in one area of Turkey is presented. The research questions are discussed and validated with the help of senior software managers (more than 15 years' experience) and then used for interviewing a variety of medium and small scale software companies in Ankara. Observations show that there is a common reluctance/lack of interest in utilizing measurements/metrics despite the fact that they are well known in the industry. A side product of this research is that internationally recognized standards such as ISO and CMMI are pursued if they are a part of project/job requirements; without these requirements, introducing those standards to the companies remains as a long-term target to increase quality.

Keywords: metric; measurement; small-medium size enterprise; ISO; CMMI

1 Introduction

It is an established fact that software metrics play an important role in ensuring the quality of software products. However, it is also observed that many software companies are not implementing any software metric programs in their organisations, as those programs are suggested. Further, the studies reveal that more than 70% of software products were developed in small- and medium-scale software companies [1]. There may be more than one reason for this; as a start we look at the definitions: in 1990, IEEE closes a gap in defining measurement standard by the following definition: "a standard that describes the characteristics of evaluating a process of product" [2]; following that, in about less than 10 years, IEEE [3] defines measurement as “the act or process of assigning a number or category to an entity to describe an attribute of that entity.” Those definitions are not controversial, nor they are misleading; however, they indicate that measurement in software engineering may not be always objective as it is different to other established branches of engineering. It is because each branch of engineering is based on basic fundamental principles of physics, but it is not so straightforward up to now to establish principles and rules for measuring software. The ongoing researches in establishing such fundamental rules and large amounts of different type of measurement techniques reported in the literature support the nature of this challenge.

The variety in measurements is induced by subjectivity, which is partially due to aiming to measure the quality of software, such that, there are software metrics, which are measure for quality attributes. Software quality metrics [3] can be treated as functions whose inputs are software data and whose output is a single numerical value that can be interpreted as the degree to which the software processes are a given attribute that affects the software quality. Further quality attributes include maintainability, flexibility, testability, usability, integrity, efficiency, reliability, correctness, interoperability, reusability and portability, which are closely related to software metrics [4].

Software metrics have potential roles at different scales in all type of information systems. Most of the companies, regardless their type, such as in government, banking and finance, education, transportation, entertainment, medicine, agriculture, and law sectors, all use software products. Naturally, maintenance (corrective, adaptive, perfective, preventive) is required for each software system on a regular/irregular basis. In such activities, the software metrics can be used for assessing the proposed modification and improvements in software systems.

Defining the problem

Metrics not only help us to evaluate a system, but they also give us ideas that help us in decision making, and they can be utilized for scheduling and planning, and for estimating costs. What we have represented so far led us to think about the practical uses of the software metrics in the industry. It is a common observation that normally large companies take initiatives to achieve quality objectives, but small and medium sized enterprises (SMEs)/companies may put quality on a lower rank, even if it is not explicitly said so. If we compare the ratio between large and small companies, especially for software companies, small and medium enterprises are dominant in producing software products when compared to large companies. For example, 77% of the software companies in Germany in 2000 were small scale [1]. Similarly in Brazil, 69% of the software companies were small scale in 2001. This data supports that most of the software products were produced by small and medium scale software companies. Hence, software companies of small/medium size are not to be underestimated and addressed separately.

The following are the research question and its sub-questions, which motivate us to make an effort to study this issue.

The research question arises: Are metrics and other tools used by the SMEs companies for achieving quality in their products?

The sub-question are as follows: If they (or some of them) are using them, to what extent are they using them? (1)

Are they using throughout the software life cycle, starting from the requirements stage through to the deployment of the software; or are they only using them for reviewing purposes (software inspection/review) and testing? (2)

2 Literature Survey: Measurement and Metrics in Software Engineering

The term *software engineering* was first defined by Fritz Bauer[1] in the 1960s "as the establishment and use of sound engineering principles in order to obtain economical software that is reliable and works efficiently on real machines". Since then, software engineering has been widely accepted as an engineering branch, and measurement has been seen as an important aspect of it, as there is an engineering principle that measurement is a mandatory task [5]. With the popularity of this branch, ongoing research is continuing on this topic with the following examples. In [6], the authors argue that measurement information should be properly processed and classified in order to provide a "better" overview of the current situation. Another example is reported in [7], where the practical problem of the applicability of measurement theory in software engineering is underlined, despite the fact that measurement theory gives a scientific base. In that work, the author discusses the challenge to propose a measurement theory for software engineering, and they approach the problem by coining the concept of weak measurement theory to solve the basic problem for the applicability of measurement theory in software engineering. Wang [8] has also attempted to apply measurement theory in software engineering. However, his work is not complete and we could not find the extended works on Morasca's nor Wang's on measurement. Related to measurement theory, Kaner has proposed a framework for the practical evaluation of software metrics in [9], which provides a more formal approach to the other existing ones.

[1] From the memoirs of Brian Randell, editor of The 1968/1969 (first) NATO Software Engineering Conference

In addition to those articles, there are several books in literature devoted to measurement theory. It would be beneficial to quickly visit them as we refer to them later in the paper.

The first book on the measurement of software was, to our knowledge, introduced by M. H. Halstead [10] in 1977. This work was bookmarked as a theory of software science, and it established the first analytical laws of computer software. In his proposal, Halstead developed quantitative laws using a set of primitive measures. Halstead's measurements are considered interesting because they can be applied after the design or completion of code. After that publication, successive books on software measurement are reported in the literature. For instance, 'A framework on software measurement', [11] by Zuse takes the theoretical, practical and evaluative view of software measurement. This book investigates software measurement principles and provides the proper guidelines for software measurement. The author evaluated all the existing measurement proposals for software at the time of writing and pointed out their pros and cons and their practical applicability to problems and, accordingly, he suggested a "proper" way of measurement. In addition to measurement, metrics are also researched in the literature; for example, 'Software Metrics' by Fenton and Pfleeger [12] is devoted specifically to software metrics; in the work, the authors discuss measurement in a comprehensive way, from the basics of measurement theory to its applicability to software engineering, which is required for software development. They explain the fundamentals of measurements and experimentations in general and software engineering measurements. Furthermore, the authors emphasize planning for measurement programs, measurement in practice, and metrics tools.

Among the popular recent books, 'Software Engineering, a practitioner's approach' [13] by R. S Pressman, is one of the base books in software engineering. This is a book which can be treated as a bible in software engineering, as it discusses many aspects of software engineering; without going into too much detail, those are most of the facets of software engineering starting from software process to the latest software development practices. By taking the measurement as the key element in engineering process, the author reports his work on applying different measurement techniques through examples. In addition, two specific chapters are devoted to software product and process metrics in the book, which are used for different languages, stages, applications and types of development.

Another recent example is Software Engineering by Sommerville [4], which is also a valuable contribution, as it provides different measurement techniques at different stages of the software and for different applications. Similar to Pressman's book, this book is not limited to specific measurement techniques. A remarkable detail of the book is, for example, that it proposes reliability metrics[2].

[2] Chapter nine: critical system specification

In some of the books, practical applications are discussed, and they provide valuable knowledge through the experiences of applying software metrics; ‘Software process improvement: metrics, measurement, and process modeling’, [14] edited by Haug, et al. is one of them. This book is devoted to reporting authentic applications of measurements and to analyzing measurement techniques, which are applied to software process improvement. As experimental data, the authors have collected a set from the European Experience Exchange (EUREX) project sponsored by the European Systems and Software Initiative for Software Best Practice in Europe.

Those books are among the most famous examples that deal with software measurements and metrics; they are widely accepted in the software engineering community. The discussion on measurement in software engineering is not limited only to those examples that we have visited briefly; it still continues. Those examples show that there have been many metrics proposed for different purposes in the software engineering domain; there are works such as [15] and [16], aiming to compile already proposed metrics. At the same time, particular implementations of measurement techniques for improving quality in small and medium scale organizations are in limited number to our knowledge.

3 Definitions of Measurement and Measurement Standards

Before we go further, we would like to define metric and measurements definitions that we adopt in this work.

A metric is formally defined as “a quantitative measure of the degree to which a system, component, or process possesses a given attribute” [2]. A “measurement” is then a task which computes a metric from the attributes of the entities within a given domain, using clearly defined rules [5]. Metrics must be purpose-oriented [17] and have clear objectives [18].

With the examples we have given so far, the role of measurement in software engineering proves itself as an essential to understanding software processes. In parallel to this claim, Bourque and colleagues [5] argue that software engineering without measurement would be hard to interpret, because without measurement, management would be difficult. According to [17], measurement is essential to monitor, understand and improve software processes as well as products and resource utilization. While those points are given credit, there are other researchers (e.g. Basili [19]; and [15] and [17]) who point out that there is a lack of consensus around software measurements. In fact, many metrics have been defined which are not used, according to [17]. While metrics need to be goal- or purpose-oriented, a goal must first be determined, along with a way of measuring the degree of attainment of the goal, and both tasks may be subjective. For example, counting the “lines of code”, as a metric, may serve to as an indication of the complexity of a system. However, line count is not a measure that provides any insight into the activity of a system, as not every line of code has the same relevance at run-time. The number of lines of code has also been claimed to be inappropriate for component-based systems; rather, complexity metrics for such systems should be based on number of components and interactions among them [20], [21].

Specifications of the rules for the process of quantification may also be ambiguous [17]. For instance, the implementation of the same metrics in different software tools to support assessment of software design has been found to give different results [22].

Furthermore, there are some software attributes that are a challenge for measurement in IT domains at various levels. For example, the elements in the IEEE standard concerning the evaluation of productivity are broad-ranging and dynamic, such as documents per person per hour or lines of codes produced per day [23].

Another important topic is productivity and its assessment. Metrics connected with productivity of IS have been controversial [24], [25]. It has been argued that traditional metrics of input versus output can work “... as long as computers allow firms to produce more of the same product at lower costs...” [26]. Such measures of productivity concentrate on the efficiency and effectiveness of the systems [27]. Overall performance measures include operational performance, especially system availability and throughput (that is, producing the output within specified time boundaries, and the quality of the content of output). However, the benefits of IT may not always be easy to measure as they can be in forms such as customer service or convenience, which may be intangible. Hitt underlines the contribution of IT to business productivity but claims there has been mis-measurements of output [28]; for instance, where customer service or convenience are the output, there are difficulties, as well as subjectivity, that may lead to mis-measurement.

An early summary of what we have reviewed so far tells us that there are no generally accepted metrics for many qualities, such as class cohesion in software development, which address software quality when new features are added; this prediction is supported in the literature (e.g. [29]). It has also been argued, in the case of software complexity, that measures are not only subjective, but that they do not satisfy a theory of measurement [30], and this charge can be laid against many IT measures.

In the end, the literature pushes us to question if it is possible to objectively measure all useful qualities of software. There are approaches to this problem. Attributes have been divided into categories of external and internal, according to whether they are indirectly or directly measurable, respectively [31]. External metrics are most likely to be subjective. More explicitly, internal attributes, such as defects, can be measured, for example by counting, while an external attribute, such as maintainability, can be measured only with the help of internal attributes which act as surrogates, such as measuring modularity with a count of components. Another internal metric is "lines of code", which is simple to implement by counting, whereas "effort" required producing those lines is difficult to determine and so is an external attribute which can only be approximated by surrogate measures, such as "development time".

There is a great deal of literature on the measurement of external attributes in software development, software quality and software maintenance [31]. Many of the metrics proposed in the literature are directly or indirectly related to structural connections between the number of classes, the number of times a class is invoked, and class size, which are all internal attributes. These measures are used as surrogates in measuring external attributes such as how flexible or reliable a system is.

Another important aspect of the software products is maintenance. Measurement and metrics are important for assessing proposed maintenance activities in software systems. As the business requirements change over time, further maintenance activities are performed. Those activities are not limited to changes in the hardware of systems but can also be change in the code, which may create a risk of instability in the system; this degradation is referred to as code decay, which is the decrease of the quality of the code due to further modifications [32], but the degradation of systems needs to be measured through the observation of activities required to add new functionalities or new hardware, or repairing faults. An indirect measure of system decay proposed in [33] involves the relative effort, time to complete, and quality of modifications. To quantify the effect of aging in operating system resources, various metrics such as "estimated time to exhaustion" have been proposed in an attempt to develop proactive techniques to limit system crashes. The "time to exhaustion" metric suggested in [34] is based on the slope estimation of the resource usage on UNIX variants and can be applied to different system resources including free memory and file table size. Another approach [35] focuses on estimating resource exhaustion through time series analysis, where they create artificial workload to the web server and monitor the resources for applications involving web servers.

When it comes to the reliability, it also requires measurement, as software quality is strongly tied to it. Reliability is the ability of a system or component to perform its required functions under stated conditions for a specified period of time as defined by IEEE [2]. The literature on reliability measures is not newly emerged. One of the earlier works on software reliability measures identifies mean time to failure and cumulative execution time as surrogate measures. [36] and [37] proposes assessment techniques based on errors remaining after the testing phase, as well as on failure and hazard rates. Those errors may be captured later via user feedback; meanwhile, errors and failures remain the main elements in measuring reliability (e.g. [38], which is a revised version of [39]) although inclusion of measures of software complexity, test effectiveness and the consideration of the complete operating environment have been recommended to make reliability assessment more accurate [40].

Coming back to the subjectivity of the measurement in software engineering, we remark that Fenton identifies reliability, maintainability and productivity as the quality attributes of high level software, and he notes that maintainability and reliability are attributes of the software itself, whereas productivity is an external attribute associated with people (the organization) dependent on processes and software [18]. He claims that the use of internal software attributes to measure these external attributes remains subjective. For example, modularity may or may not be considered a surrogate for maintainability. The metric of class number may be taken into account more when modularity is considered, because the increased number of classes allows for greater precision in expressing dependencies [41].

By definition, metrics have been developed to measure aspects of software development. For example, productivity of a development team provides a measure of delivery of maintenance activity [42]. Modularity and a system's resulting flexibility are important for further maintenance. As with other software metrics, the objectives of a software development project shape the criteria for their evaluation. In some environments, speed of development is critical, in which case a low number of classes may be desirable because the development team is rushing to produce software within a tight time frame. A related adverse by-product may be that the production rate for lines of code per day is high because of the duplication of code elements. Conversely, when future maintenance is considered to be important, the metric of class number may be useful because the increased number of classes allow for greater precision in expressing dependencies [41], despite a lower production rate of lines of code per day because more design thought is put into the software construction.

As seen from this quick review of the literature, the topic of measurement/metrics is not a narrow topic and one must comprehend the details of the metric in order to employ it, including the circumstance for which it has been proposed; this makes measurement/metric proposals not always straightforward. Such complexity may discourage SMEs in the software industry, where time and human power are usually precious. The good news is that there are some internationally recognized standards in parallel to this issue.

3.1 International Standards

So far, we have provided and discussed our literature survey to do with software measurement and metrics. Through this report we can deduce that software measurement and metrics are not only challenging, but they also can be controversial, subjective and open to discussion. Despite those handicaps, there are some standards which have international reputation. For instance, the capability maturity model integration (CMMI) is presently accepted as the best accreditation for the software industry. Some of the CMMI certified organizations are Boeing's Space Transportation System software, Tata Consultancy Services[3], Telcordia Technologies[4] and Granter Inc.[5] By adopting the CMMI, automatically, they consider the best practices of measurement in their processes.

With the International Standards Office (ISO), providing ISO9001 and ISO9000-3, companies can have two more standards to certificate and authenticate their work. ISO 9001, which is a standard for any type of product, was basically not for the software industry, but its application to software was initiated by TickIT (UK), which provides the methodology for adopting ISO 9001 to the software industry. As a figure, in 2002, 1252 organizations in 42 countries were accredited by ISO 9001 (TickIT) (in 2002) (www.iso.org, www.isoqar.com/iso9001/ qualintro.htm). ISO 9000-3 explains how ISO 9001 can be applied to software. ISO 9000-3 (http://www.praxiom.com/iso-9000-3.htm) provides the guidelines which lead and serve as an all-inclusive standard for the software industry. ISO 9000-3 is used in developing, supplying, installing, and maintaining computer software. For acquiring the ISO 9000-3 certification, an organization must develop the organization's software quality assurance (SQA) team, implement the organization's SQA systems and undergo certification audits.

Standards on metrics and measurements are not limited to these examples; recalling that this paper aims to shed a light how much the measurements and metrics are adopted in SMEs, we discuss this limitation in a literature survey on standards in the last section.

4 Research Methodology

For the purposes of our research, the attributes which are deemed to be of interest in the literature on metrics are more important than the form of the metric and measurements. In seeking an answer to the research question, we have reviewed discussions on the objectivity and complexity of the applicability of the measurements/metrics. The literature we have surveyed caused us to think on whether the SMEs use measurement as a tool in their business or not. In the case that they are using them, we aim to learn about how much they benefit from them.

4.1 Research Framework

This research adopts a two-stage approach to address the research question.

[3] http://www.tcs.com/homepage/Pages/default.aspx (accessed in 2010)

[4] http://www.telcordia.com (accessed in 2010)

[5] http://www.gartner.com/technology/home.jsp (accessed in 2010)

Firstly, we formed a body-of-knowledge including software metrics and measurements. As Figure 1 shows, while reviewing the literature, we saw that an empirical study would help to fill the practical applications of the measurement/metric in SMEs in the software domain; and we identified the research question accordingly. Later, with the preparation/modification of the interview questions, we saw that approaching the research question via sub-questions would ease and increase the validity/reliability of the research.

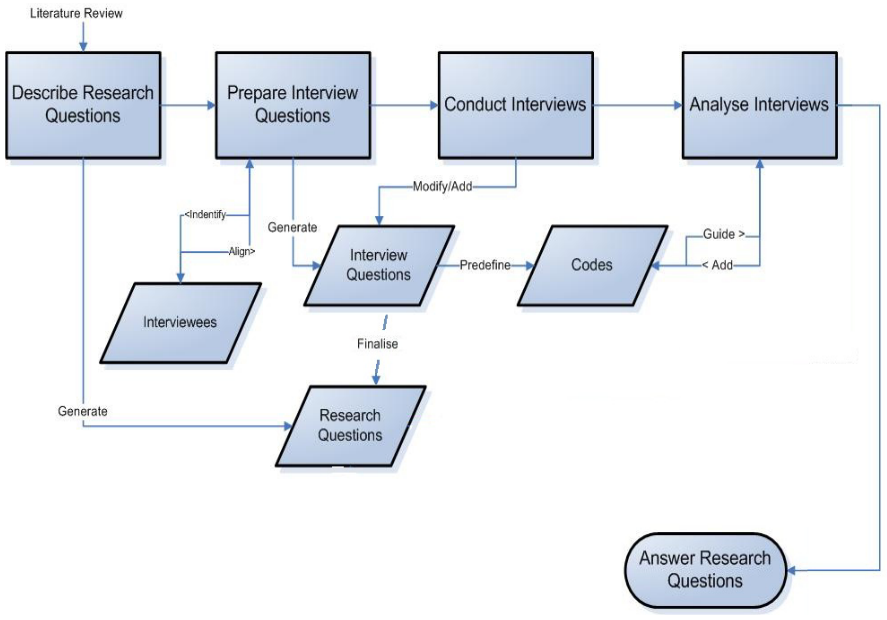

Figure 1
Research Framework

Given the qualitative nature of the research, we have selected to conduct semi-structured interviews to collect data from the field. The preliminary questions are based on the literature survey and the gaps we identified to our knowledge.

We started looking for local SMEs to approach software developer firms in Ankara as potential key informants, who later discussed the validity of the preliminary interview questions. As a selection criterion, those key informants[6] have considerable knowledge and experience on software engineering in the industry. With the fourth and the fifth informants delivering similar comments and advice, we decided to stop aligning questions and look for interviewees.

6 For privacy reasons, we do not mention the names of the key informants, nor any identifier that can reveal their workplaces.

The second stage of the research is based on the aligned semi-structured interview questions from the previous stage. Those aligned questions formed the main discussion points with the interviewees. Similarly, these interviewees were selected from different companies with the selection criterion of being senior professionals in their fields.

Additionally, we added the analyses of semi-structured interviews and ranks which we collect from the interviewees to this data pool in order to design, revise and validate our research (Figure 1). This way of doing research fits into the grounded theory of Glaser [43], i.e. research where the data collected during the research guides the research.

Before going to the interview sections we selected keywords as codes to be used while analyzing the free comments of the interviewees. Those codes accelerated our analyses to cluster quotes under similar topics.

As these steps indicate, this qualitative research is designed as a descriptive study rather than an explanatory one, and it adopts partially grounded theory and interviews as research techniques.

4.2 Collection, Analysis and Interpretation of the Results

All the questions were finalized and validated by chief executives in established software companies' in Ankara. First, we interviewed these executives and, based on these discussions and their recommendations; we redesigned our questions and sent them to be validated through e-mails. After their modification, the following set of the questions and their scope were finalized and summarized, as in Table 1.

Table 1

Survey questions and their scope

Q. No.	Questions	Scope
1	How many software professionals are you employing?	1-5 / 5-10 / 10-20 / 20-50
2	How many of those s/w professionals are working with you more than 2 years?	Stability of the employees in particular industry
3	What is the average experience of your s/w professionals not only in your place but in the industry?	Average experience of software professional
4	How many core team members are aware about the usefulness and importance of software measurements to achieve quality objectives?	Experience and awareness of team members for measurements in software Industry
5	Which tools and methods are you applying for achieving quality objectives?	Knowledge of tools for quality measurements
6	Does the company use software measurement as tool in the business?	Applicability of measurement tools for quality measurements

7	Have you got any measurement guidelines/framework for controlling/assessing your products' quality?	Awareness and availability of measurement tools
8	In terms of software development, are you using any internationally recognized standards for achieving quality objectives and to improve your business? For example, ISO, IEC, CMMI.	Availability and applicability of international standards
9	If you already have any of those standards, did they help you to improve your company up to your expectations? If so, in what aspects?	Effect and results of using international standards
10	Do you think CMMI is a criterion to assess a company's quality and reputation?	Aim of adopting highest standard for a company
11	Which type of measurements are you using? Could you please name few of them e.g. resource management of computer, number of: line of code, loops, modules, errors...etc	Specific metric for measurements
12	Has the company got expert team or members who are software quality engineer or experienced in software measurement; if so, how many? Do you hire any person outside for this activity, alternatively?	Availability of software quality engineers
13	In the full software development cycle (from requirements, through design, development, testing, to deployment) are you using any kind of measurement?	Use of measurement Techniques in software life cycle
14	Do you give more importance to inspection or testing your products? In other words, do you do assessment while inspecting or testing?	Software review/inspection
15	Where else are you using measurements; e.g. maintenance or support to you clients?	Further use of measurement
16	Are you following quality guidelines/frameworks/ measurements while doing business with your partners?	Use/effect of measurements in business
17	Do you think that software measurement can improve the quality of your products? Please provide an example while answering.	Actual knowledge and awareness of quality objectives
18	Do you think there are additional/alternative tools/methods than software measurements in order to improve your business in software development?	Awareness of other tools for improving quality and level of company

After finalization of the questionnaire, the interviewing technique (section 4.1) is adopted for examining the applicability and awareness of software measurement and metrics in software companies located in Ankara. We collected the list of software development companies from METU-Technopolis[7], a place inside a leading university of Turkey, Middle East Technical University, where the offices of approximately 280 companies are located. Of those companies, more than 90% are SMEs operating in the ICT (60%) and electronics (25%) industries. As we mentioned earlier, most software is developed in small/medium-scale companies. This practice is more common in developing countries, but it also found in developed countries such as in Germany. We considered companies which between 5 and 50 software developers as small- and medium-scale companies. We aimed to interview only those companies in this category; hence we visited about half of the listed software companies.

5 Observations

We have a general interpretation of the results that shows at the first glance that SMEs are not inclined to put the measurement at the first rank, which is in parallel with [17] what has been discussed previously. Despite the fact that there is a considerable amount of metrics proposed in literature, these tools are not considered; some of them are not even known among the interviewed companies, although they are actively doing business with a variety of clients (from defense to accountancy).

We have not found encouraging results regarding measurement for achieving quality objective in their software programs. Most of the companies have ad hoc evaluation criteria in their software development programs. Their main aim is to complete and deliver the job as soon as possible because they are tied to strict time frames. For this reason, most of the companies failed to answer our survey questions with details, as there are no specific measurement/metrics programs implemented in their organisations. The most they do is that, in the case of failure or complaint, they try to remove the errors.

At a small set of companies, the organisers found interested in measurement and metrics. The appendix has a sample set which shows the feedback from the interviewees.

We have been informed that not applying measurement techniques is not only because of reluctance but also it is considered as “not necessary” and “not required” in their project contracts.

[7] http://www.metutech.metu.edu.tr/cms/index.php?Lang=EN (accessed in 2010)

Although the role of measurement in software engineering is given credit and still remains as one of the popular topics in the software domain, SMEs do not see any "persuasive" benefit to urge them to study, evaluate and choose measurement techniques and metrics to adopt in their workplaces. This observation may be seen as challenging what we have recovered in the literature survey e.g. [5], [15], [17] and [19]; however, feedback from relatively larger software companies shows that measurement becomes a necessity in order to monitor, understand and improve software processes along with software products and resource utilization as the companies get larger in the number of employees. The following feedback is from relatively larger companies.

> *"Definitely, yes, (s/w measurement can improve quality of the product). The outcome of the measurements can be used as input in following projects; hence, more suitable project scheduling is possible, which makes the application correct and high-quality" (interviewee 5, noe (number of employee): 20-50)*
>
> *"Yes (s/w measurement can improve quality of the product). With measurement the tasks can be planned and managed. Staff can be educated with the composite metrics" (interviewee 11, noe: 20-50)*
>
> *"Yes (s/w measurement can improve quality of the product), we identify spots to rehabilitate and we take preventive actions with the aid of measurement" (interviewee 13, noe: 80)*

Pretty much all of the companies informed us that the quality of the products should be assessed by the developer, the team leader and/or through meetings for software inspection/review. However, to achieve quality standards, none of the interviewees has put any metric or measurement techniques forward. In a broader sense, most of them do not use any measurement tools in their business, except some who limit measurement to evaluating jobs in price:

> *"…we calculate e.g. 33 hour requirement for the client" (interviewee 17, noe: 5-10)*

Because quality is strongly tied to measurement in software products, as we have surveyed in the literature, we attempted to collect more information about the quality standards from the interviewees. The result is that the following internationally recognized standards (such as ISO, CMMI) are beneficial in general; however, adopting such methodology requires time and patience:

> *"An increase in quality but slowing development due to procedure" (interviewee 4, noe: 1-5)*

Another observation is that even simple metrics such as lines of code are open to discussion:

> *"Yes, (s/w measurement can improve the quality of the product) but cannot be single criterion alone; e.g., the number of LOC was 200 in a*

program we wrote 10 years ago and the performance was poor. Later, we reduced it to 3 LOC and it runs correctly and fast. Here, less LOC brought an advantage through speed; however, higher LOC may not be a disadvantage; at the same time, it should run correctly. Another example is that we have delivered a project with 10 forms although we have been contracted for one form. Here, some of the forms were simple while the others were a separate project each. Those numbers became important while negotiating on price" (interviewee 16, noe: 10-20)

Obtaining an internationally recognized certificate is not an easy task. However, those certificates are not always obtained because the company would like to make the work place "better" and/or up to a standard, but rather because they are required in project specifications. While discussing the role of CMMI we collected the following feedback:

"CMMI is a very important criterion but not sufficient alone. The course could be left after obtaining CMMI" (interviewee 5, noe: 20-50)

"… it (CMMI) may stay as a label and not be applicable logically and efficiently for small companies" (interviewee 6, noe: 10-20)

"CMMI cannot always be followed; a company can flex it according to internal dynamics" (interviewee 14, noe: 10-20)

6 Results, Discussion and Recommendations

Recalling our survey of discussions on the absence of consensus on software measurements [15] [17] [19] [22] we predicted a reluctance to use metrics in the industry. This is gets more complicated with controversial and subjective proposals in productivity measurements [24] [25] [28]. Similarly, maintenance is seen as one of the most important activities in software systems, but indirect measurements provide subjective and system-specific solutions [33] [34] [35]. Also reliability, hence error-failure measurements, are important [36] [37] [38]. In addition to these, assessing the developer team [42] is another measure. Coming along with the metrics and measurements, popular international standards (CMMI, ISO9001 and ISO9000-3) stand as common criteria to maintain quality levels in software companies.

6.1 Results

As summarized and clustered above, we prepared our interview questions (section 4.2) to address the motivation for measurement and tools to measure productivity, reliability, maintenance, developer teams and to address the awareness of international standards.

We concluded the following results in conjunction with our observations (section 5):

Measurement is not a priority unless it is money-oriented (1)

For small companies, until they get some financial benefit/support, they do not implement any specific measure to improve quality.

The use of measurement is limited in the assessment of software (development) quality and it is considered a long term activity (2)

Most of the SMEs have the perception that software measures are only used for improving quality, but that it requires a long time to implement in the workplace.

Measurement and metrics are limited due to the unawareness of measurement techniques amongst the developer (3)

In fact, there is a considerable confusion about what the measurement activities are for in improving quality of the product. They know the fundamentals, that a code should be reviewed and metrics should be applied, but not which specific tools and techniques should be applied at different stages of software development; most of them are not aware or interested. This is closely tied to result (1) as financial benefits are seen as main motivator in the industry.

The use of software metrics is limited due to heavy time pressure for the delivery of products (4)

This is also a hard truth for the software industry and especially for the SMEs, who are considerably affected in achieving quality objectives due to heavy time pressure, as they are often working on projects with tight timeframes.

The use of software metrics is limited due to lack of highly experienced professionals in the company (5)

In SMEs, there are several constraints, including (and maybe led by) financial constraints. To achieve quality objectives, any company must have experienced professionals in permanent positions or must hire them for some specific activities, e.g. software inspection/review. However, financial constraints are a barrier to doing so. Further, changing organizations amongst software professionals is not an uncommon practice; when software developers gain some expertise in a specific area, they get offers from bigger industries with better packages; hence, it is not uncommon for small companies to lose those employees who become experienced in evaluating measurement and metrics..

The uses of measurement techniques are limited due to an unawareness of the depth knowledge of quality issues in the software development process (6)

Before joining the software industry, most, but not necessarily all, professionals come from universities with an engineering degree. However, in most of the syllabi of engineering branches, quality issues are not given emphasis in the course curriculum. Even in computer engineering, software quality management is not an essential part of the study curriculum unless the student chooses to take such elective courses.

An obtained standard or certificate may be used just as a label (7)

It is not uncommon to require standards such as ISO or certificates such as CMMI as a prerequisite in project specifications. In order to have a chance of entering the pool of companies tendering for projects, companies are motivated to apply such standards/certificates. However, after getting involved in projects, the certificate may stand on the wall and the company does not necessarily follow its directives.

6.2 Recommendations

This paper presents our survey of SME measurement activities used to achieve quality objectives in their software products. Although improving the quality of software seems to be a prime objective in the industry, our survey reports that most SMEs do not spend as much care as is encouraged in the literature. This study also hints at the effects of an absence of consensus regarding software measurements and, as a result, an associated reluctance to use metrics in the software industry. On the other hand, neglecting quality objectives bears the risks of delivering low quality software; obvious consequences are not only the rejection of the projects but also a poor reputation for the software company, an important element in the long term for any developing company.

Apparently, the bringing of the metric/measurement notion into a workplace may increase budgets for projects and/or reduce short-term earnings because adopting a notion in a company requires stability (keeping adoption with changing employees and projects) and separate documentation for knowledge management for further projects to apply similar measurements. However, our study supports the view that it is not only our suggestion that SMEs adopt the measurement and metrics in their software development program, but also that those companies give credit to this practice.

Keeping in mind that this paper has limitations while focusing on the application of quality methods specifically, the reader should be informed that this report should be read in conjunction with related literature on quality in the software domain. Software quality issues include the application of the measurement methods, but it is not limited to this; for example, while getting into more technical detail, the quality of the applied algorithms and program code are given credit generally in the literature. Recent examples include [44], where the author underlines the performance linked to those two items while developing software products.

Conclusion and Future Work

As following discussion of results and limitations indicate there is room to research to itemize the reasons linked with the findings of this study.

Adopting metrics/measurements in SMEs is not an easy task, as we mention in Section 2. Hence, blaming those companies for not doing so would not contribute to a solution and would leave the recommendation unsupported. Rather, proposing a way to adopt metric/measurement applications could encourage SMEs to get motivated in this topic. For such an attempt, a framework of IT, project management and economics may generate a method of approach to introduce the idea of metrics and measurement within SMEs in a long-term, step-by-step approach.

When proposing such a method, the limitations of the results presented in Section 6.2 should be considered. A general limitation is that we have conducted interviews locally. However, this limitation is not a great constraint as the companies present a broader variety of interest as we mention in Section 5. Another one is that we had only one person per company to interview. Most SMEs have a limited number of employees; for this reason we do not expect any considerable variety of information within a company. However, obtaining information about the employees' degrees and their course curricula could extend results (3), (5) and (6) as a more focused questionnaire could be prepared to investigate their knowledge on metrics and measurements.

The current study did not have the chance to study project requirements in order to analyse the details of results (1) and (4). We are aware that as a part of the industry, there are many companies working on delivering bespoke information systems. However, we have excluded this issue in this study. This limitation opens an associated and further study on project-based investigation in SMEs.

The current study gives signals that although some companies have acquired internationally recognized certificates and standards, they may not follow them, as summarized in result (7). As we have observed, standards may be used only as labels. We see the potential for future studies focused on revealing more concrete reasons for delaying obtaining these standards.

Appendix: Sample set from the interviews

1	2	3	4	5	6	7*	8	9	10	11	12	13	14	15	16	17	18
10,20	12	4	4	CMMI based	no	a	ISO, CMMI	Extraordinary change in SE. Especially while having a project in defense. As a result of std., things are getting easier like repeating, inspecting and adding.	yes		1 quality eng.	Sub-versioning is used related to configuration management	Inspection is important, testing is still on the developer	Measurement in maintenance	When we are sub-contractor, yes.	No programming measurement is used, yet.	
1,5	1	1,4	all	No time to apply	No time to apply	d	Not yet		yes		no	Eclipse IDE and integrated SVN	testing		no	Definitely yes	
1,5	1	1,5	2		yes	a, b, c	ISO	Increase in quality but slowing in development due to procedure	no	Resource management, loop, module, error counts	1	We use our program developed here	Both in inspecting and testing	Maintenance and support to client		More quality products thanks to minimized errors; less problems with the clients	
20,50	10	3,4	all	ISO, CMMI	Screen, code and sql line count	b, c	We use ISO and working on CMMI	Yes, increase in quality, software reverse ratio increased, code library is more productive	CMMI is a very important criterion but not sufficient alone. The course could be left after obtaining CMMI.	Code line, error, database table counts	2 software quality eng.	MS TeamSystem, Foundation Server	More on the inspection phase		Yes, quality guidelines and document templates	Definitely yes. Outcome of the measurements can be used as input in following projects hence more suitable project scheduling is possible, which makes the application correct and quality.	
5,10	3	1,3	all	ISO 9001:2000, tools and methods defined within our quality management.	TS 12207	a, b, c	ISO		yes				both	Customer support	yes		
5,10	6	7,8	1	We don't use	We don't use	a	No		Yes	We don't use	No	No	Test	We don't do measurement	No	It may without doubt	No comment
1,5	2	10	1	Source controlling, regular testing	No	a, c	No		No	Source management, error count	2, we provide this service, too	No, we develop ourselves	Test				Unit testing
20,50	10-20	8	We have ISO; majority knows	We follow ISO quality standards	We don't use tools	c	We follow ISO quality standards	Yes, it made us to work more productive and in an order.	Certainly it is an important criterion	LOC, # of modules, ratio of compile time error to runtime errors	1	Enterprise Architect	Both of them are important equally for us.	We don't have currently.	Yes	It increases quality of the products.	
20,50	35-40	5	15	Agile, Atlassian JIRA, Continuous Integration	JIRA	a, b	ISO, SPICE	Beneficial indirectly. Measuring and time frame provide essential benefits	No	We have non-integrated solutions (see 13)	5-6. we have consulting firm working of quality.	Total lines, LOC, comment lines, DP, DP/LOC (see 11)	Testing and continuous Integrations are more important	We do measurement on every field, including support and sale.	Yes	Yes, with the measurement the tasks can be planned and managed. Staff can be educated with the composite metrics.	Continuous Integration time frame. Testing automation .
10,20	6	4	all	We follow approach compatible to CMMI	no	c	Because the company is small we don't have any certificate but we try to follow standards e.g. ISO	-	CMMI cannot be followed always; company can flex it according to internal dynamics	We follow RUP life cycle. We test speed and security of the program	3 people	Enterprise Architect	In both	During instalment to the client	yes	Yes, we ensure re-usable codes to save work power	-

* (a)The developer himself checks the quality with available tools (b)The team leader checks the quality regularly (c)Meetings are organized to evaluate the quality of code, (these meetings are called software inspection or software review) (d)The company only bothers about the output i.e. if programs produce the output without any bug or error, no matter how the code is built (e)Any other

References

[1] Wangenheim, C. G. v., T. Punter, and A. Anacleto. Software Measurement for Small and Medium Enterprices - A Brazilian-German view on extending the GQM method. in 7th International Conference on Empirical Assessment on Software Engineering (EASE) 2003, Keele, UK

[2] IEEE, IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries - 610. 1990: IEEE 217

[3] IEEE, IEEE Standard for a Software Quality Metrics Methodology, in IEEE Std 1061-1998. 1998. document number

[4] Sommerville, I., Software Engineering. 7th ed. 2004: Pearson Addison-Wesley. xxii, 759

[5] Bourque, P., et al. Lack of Consensus on Measurement in Software Engineering: Investigation of Related Issues. in 14th International Workshop on Software Measurement IWSM/MetriKon. 2004. Magdeburg, Germany: Springer-Verlag

[6] Gómez, O., et al. A Systematic Review Measurement in Software Engineering: State-of-the-Art in Measures. in First International Conference on Software and Data Technologies. 2006, Setúbal, Portugal: Springer-Verlag

[7] Morasca, S. Foundations of a Weak Measurement-Theoretic Approach to Software Measurement. in FASE 2003. 2003. Warsaw, Poland: Springer-Verlag

[8] Wang, Y. The Measurement Theory for Software Engineering. in Canadian Conference on Electrical and Computer Engineering (IEEE CCECE) 2003. Montreal: IEEE

[9] Kaner, C. and W. P. Bond. Software Engineering Metrics: What Do They Measure and How Do We Know? in 10th International Software Metrics Symposium (Metrics 2004) 2004. Chicago, IL

[10] Halstead, M. H., Elements of Software Science. Operating and programming systems. 1977: Elsevier Science Ltd. 142

[11] Zuse, H., A Framework of Software Measurement. 1998: Walter de Gruyter. 755

[12] Fenton, N. and S.L. Pfleeger, Software Metrics. 2nd ed. 1996: Pws Publishing Company. 638

[13] Pressman, R.S., Software Engineering: A Practitioner's Approach. 5th ed. 2001: McGraw-Hill Science

[14] Haug, M., E.W. Olsen, and L. Bergman, Software process improvement: metrics, measurement, and process modelling. 2001: Springer 391

[15] Purao, S. and V. Vaishnavi, Product Metrics for Object-oriented Systems. ACM Computing Surveys, 2003. 35(2): pp. 191-221

[16] Herlocker, J. L., et al., Evaluating Collaborative Filtering Recommender Systems. ACM Transactions on Information Systems, 2004. 22(1): pp. 5-53

[17] Berander, P. and P. Jönsson. A Goal Question Metric-based Approach for Efficient Measurement Framework Definition. in ISESE '06: Proceedings of the 2006 ACM/IEEE international symposium on Empirical software engineering. 2006. Rio de Janeiro, Brazil

[18] Fenton, N., Software Measurement: A Necessary Scientific Basis. IEEE Transactions on Software Engineering, 1994. 20(3): pp. 199-206

[19] Basilli, V. R., G. Caldiera, and H. D. Rombach, Goal Question Metric Paradigm, in Encyclopedia of Software Engineering. 1994, John Wiley & Sons. document number

[20] Gill, N. S. and Balkishan, Dependency and Interaction-oriented Complexity Metrics of Component-based Systems. ACM SIGSOFT Software Engineering Notes, 2008. 33(2)

[21] Mahmood, S. and R. Lai, A Complexity Measure for UML Component-based System Specification. Software: Practice and Experience, 2006. 38(2): pp. 117-134

[22] Lincke, R., J. Lundberg, and W. Löwe. Comparing Software Metrics Tools. in Proceedings of the 2008 international symposium on Software testing and analysis. 2008

[23] IEEE, IEEE Standard for Software Productivity Metrics, in IEEE Std 1045-1992. 1993. document number

[24] Du, J., X. Yang, and Z. Wang. Effective Runtime Scalability Metric to Measure Productivity in High Performance Computing Systems. in Conference on Computing Frontiers. 2008. Ischia, Italy

[25] Hitt, L. M. and E. Brynjoifsson, Productivity, Business Profitability, and Consumer Surplus: Three Different Measures of Information Technology Value. MIS Quarterly, 1996. 20(2): p. 121

[26] Brynjolfsson, E. and L. M. Hitt, Beyond the Productivity Paradox. Communications of the ACM, 1998. 41(8): pp. 49-55

[27] Scudder, R. A. and A. R. Kucic, Productivity Measures for Information Systems. Information & Management, 1991. 20: pp. 343-354

[28] Hitt, L. M., Economic Analysis of Information Technology and Organization, in Sloan School of Management. 1996, Massachusetts Institute of Technology. document number: 165

[29] Mäkelä, S. and V. Leppänen. A Software Metric for Coherence of Class Roles in Java Programs. in 5th International Symposium on Principles and Practice of Programming in Java. 2007. Lisboa, Portugal

[30] Misra, S. and H. Kilic, Measurement Theory and Validation Criteria for Software Complexity Measures. ACM SIGSOFT Software Engineering Notes, 2007. 32(2): pp. 1-3

[31] Vliet, H. v., Software Engineering: Principles and Practice. 3rd ed. 2008: John Wiley&Sons. 740

[32] Parnas, D. L. Software Aging. in 16th International Conference on Software Engineering (ICSE-16) 1994. Sorrento, Italy: IEEE

[33] Eick, S. G., et al., Does Code Decay? Assessing the Evidence from Change Management Data. IEEE Transactions on Software Engineering, 2001. 27(1)

[34] Garg, S., et al. A Methodology for Detection and Estimation of Software Aging. in The Ninth International Symposium on Software Reliability Engineering. 1998. Paderborn, Germany

[35] Li, L., K. Vaidyanathan, and K. S. Trivedi. An Approach for Estimation of Software Aging in a Web Server. in International Symposium on Empirical Software Engineering. 2002

[36] Musa, J. D. The Use of Software Reliability Measures in Project Management. in Computer Software and Applications Conference, 1978. COMPSAC '78. The IEEE Computer Society's Second International. 1978

[37] Yamada, S., S. Osaki, and Y. Tanio, Software Reliability Measurement and Assessment Methods During Operation Phase and Their Comparisons. Systems and Computers in Japan, 1992. 23(7): pp. 23-34

[38] IEEE, IEEE Std 982.1 - 2005 IEEE Standard Dictionary of Measures of the Software Aspects of Dependability, in Revision of IEEE Std 982.1-1988. 2006, IEEE. document number

[39] IEEE, IEEE Standard Dictionary of Measures to Produce Reliable Software - IEEE Std 982.1-1988. 1989. document number

[40] Xu, S. An Accurate Model of Software Reliability. in 13th IEEE International Symposium on Pacific Rim Dependable Computing. 2007

[41] Bar, H., et al., The FAMOOS Object-oriented Reengineering Handbook. 1999. 329

[42] Stark, G. E., Measurements to Manage Software Maintenance. 1997: http://www.stsc.hill.af.mil/crosstalk/1997/07/maintenance.asp (checked in 2009). document number

[43] Glaser, B. G. and A. L. Strauss, The Discovery of Grounded Theory. 1967: Aldine de Gruyter. xiv, 271

[44] Keszthelyi, A., Remarks on the Efficiency of Information Systems. Acta Polytechnica Hungarica, 2010. 7(3): pp. 153-161