Educational Issues in the Teaching of Software Measurement in Software Engineering Undergraduate Programs

Mónica Villavicencio[1,2]

[1]École de technologie supérieure, Montréal, Canada
[2] CIDIS-FIEC, Escuela Superior Politécnica del Litoral (ESPOL), Guayaquil, Ecuador
mvillavi@espol.edu.ec

Alain Abran[1]
[1]École de technologie supérieure
Montréal, Canada
alain.abran@etsmtl.ca

Abstract— **In mature engineering disciplines and science, mathematics and measurement are considered as important subjects to be taught in university programs. This paper discusses about these subjects in terms of their respective meanings and complementarities. It also presents a discussion regarding their maturity, relevance and innovations in their teaching in engineering programs. This paper pays special attention to the teaching of software measurement in higher education, in particular with respect to mathematics and measurement in engineering in general. The findings from this analysis will be useful for researchers and educators interested in the enhancement of educational issues related to software measurement.**

Keywords— measurement; software engineering; higher education

I. INTRODUCTION

The software engineering term was coined more than 35 years ago [1] and it is still considered in an immature stage [2]. Authors in software engineering domain generally agree that the software industry is still facing the same and recurrent problems, including deficiencies in software quality, difficulty in predicting and managing risk, exceeding time and budgets of delivering projects, among others [1, 3, 4]. A contributor to this situation may be that current software developers have not been properly educated in the software engineering field. Several authors have contributed to the improvement of software engineering education through their studies on the areas and topics that should be taught in university related programs. For instance, Parnas in 1999 emphasizes the need for treating software engineering separate from computer science and as an independent engineering discipline [5]. According to him, universities should focus in preparing undergrads to acquire the required knowledge and skills to work immediately after graduation. From this perspective, the author maintains that engineers face the challenge of producing reliable and safety software. To do so, they must know and apply mathematics and science with particular relevance in computer science.

Other authors posit that students in this domain should be educated as engineers [6]. This means that students should learn mathematics, science, engineering principles and should have learned mastering the theoretical foundations, design methods and technology and tools of the discipline [6]. In addition, they also suggest the development of skills, such as communication, self-learning, problem recognition and managing, among others [6]. The application of engineering principles in software engineering education has also been highlighted by Hawthorne and Perry 2006 [7] by emphasizing the importance of applying them in unfamiliar contexts. Complementing this view, Van Vliet 2006 [8] points out that the human and social dimensions should be acknowledged for its relevance in the educational process.

From the above discussion, math and engineering principles are considered essential in software engineering education. Hence, they should be considered as fundamental topics in the software engineering curricula. It is important to remark, however, that measurement, being one of the engineering principles, has received relatively little attention in software engineering education. This paper explores this issue and focuses on this area of study. This paper is organized as follows. Section 2 explains what math and measurement consist of, followed by section 3 which presents a summary of educational trends in the teaching of measurement and math at universities. Section 4 presents some findings regarding the teaching of software measurement. Finally, section 5 discusses aspects to be considered in the teaching of software measurement in higher education.

II. MATH AND MEASUREMENT

Mathematics (math) and measurement are fundamental subjects taught in engineering curriculum at universities. On the one hand, the primary areas of math commonly covered in an engineering program include: algebraic structures, linear algebra, basic logic, calculus, and statistics among others. On the other hand, measurement includes subjects such as metrology, measurement technology, measurement and instrumentation which are taught in electrical, mechanical and electromechanical engineering programs [9-11].

The legitimacy of mathematics is acknowledged for its relevance in promoting reasoning [6, 12] and logical thinking; by studying math, students develop abilities to: make and test conjectures and judgments [13]; be better problem solvers; make abstract conceptualizations and generalizations [14]; and model, study and understand real-world situations [13, 15]. Measurement, on the other hand, is also acknowledged as important in engineering curricula as it enable students to develop and improve processes and products, to demonstrate their compliance with standards, and to guarantee the quality of products and services [16]. Furthermore, measurement allows people to obtain information, acquire knowledge and to describe anything accurately [17, 18].

In engineering, math and measurement are used in everyday activities. For measuring, engineers need to know about standardized base measures (i.e. mass), units (grams, kilogram) and symbols (m and kg). Engineers also use derived measures, which are a combination of basic measures. From these measures, secondary units are generated. For example, "force" is a derived measure obtained by the combination of mass, length and time. The secondary unit assigned to "force" is called "newton", which is represented with the symbol "N" and is obtained by the formula: newton = kg x m/s2. As can be noticed, derived measures are formed by using mathematical operations with different units.

Mathematics is also needed when engineers need to make calculations for measuring objects that have the same units and characteristics. For example, for measuring the perimeter of a fence, a "tape measure" can be used as a measurement instrument, which could be utilized several times. In this case, all the individual measures are to be added.

In engineering, there are also conversion factors that enable the conversion from one unit system to another.

In summary, measurement requires measurement units, unit systems, etalons (measurement standard), conversion rules, instruments and scales in order to perform measurement activities, while in math these requirements are not necessary to make calculations.

Math and measurement are similar in the sense that both use symbols and follow methods (sequence of steps) for mathematical operations or measuring entities. For example: the calculation of a root square or the measuring of the volume of a prism.

They are complementary because math is used in engineering and science for implementing measurement activities. In addition, measurement data can be analyzed by using statistics to generate, for example, estimation models.

III. Educational Trends in Teaching Math and Measurement

Every year, a greater number of youths enroll in university programs [11, 19]. Among them, it is common to observe students with different degrees of interest for pursuing their studies. This variety of students' profiles is a challenging issue to university teachers because they have to look for ways to increase the odds of leading them to reach the expected learning outcomes. Fortunately, young students are full of energy, abilities and interests that can be taken into account in the teaching and learning to involve and engage them in their learning process [22]. This is in alignment with the constructivist perspective. According to this perspective, the students' learning is characterized by their active participation in the construction of their own knowledge by discovering the answer or solution to a specific problem or situation [19-21]. An increasing use of constructivism is being observed for the teaching of relevant subjects in university programs.

Larsen and Zandieh 2008 [22], for example, conducted research to support the learning of mathematics. Specifically, they focused on abstract algebra for undergrads by using the theory of realistic mathematics education (RME). Freudenthal (as referred in [22]) highlighted the fact that math is taught in a way in which students are led to follow rules and algorithms without allowing them to reinvent math for themselves and learn [22]. Research projects, similar to the one conducted by Larsen and Zandieh 2008, are being addressed in RME-guided instructional design in other areas of math, such as geometry and differential equations at the undergraduate level [22].

In the statistics field, one study has focused its attention on university students with problems in understanding inferential statistics, such as sampling distributions, hypotheses tests and confidence intervals [23]. For overcoming these problems, Castro *et al.* 2007 [23] suggested the use of computational and non-computational assessment for identifying misconceptions in students.

Innovation in education is also an important ingredient to be considered in the teaching of measurement topics in engineering programs. Innovation, here, refers to the inclusion of new approaches in education as it is the case in the constructivism view of teaching as well as in the use of measurement instruments and tools.

Eugène 2006 [10] reports on the use of the problem-based learning approach in the teaching of basic-electrical measurement at the Université catholique de Louvain. The objective was to gradually replace traditional educational methods in which lectures are predominant and usually precede exercises and lab sessions. By using the problem-based method, student teams were asked to plan an experimental session on the classical fluorescent tube and evaluate its appropriateness to dim a lamp from an economical perspective. The experiment also included an evaluation of the consequences on visual comfort and pollution. This example is an illustration of how an active learning approach can be implemented in the measurement domain.

In the same line of promoting active learning in the teaching of measurement, Szarka 2010 [11] explains how she matched the Laboratory of Measurement Technology practice with real life problems in order to catch the students' attention. According to Szarka, the way in which a problem is presented to students is crucial for their motivation. To illustrate the approach, she described two examples of the same problem. In the first example, the problem to be solved was explained with more scientific description; while the second was presented by describing an everyday life application. Examples of experiments included: measurement and analysis of the mobile-phone's vibration; and measurement with laser distance sensor. Students had to work in teams, develop a plan, define tasks for each member along with time schedule and deadlines; and make a public presentation of their solution. Students had unlimited access to a remote controlled laboratory available in two languages, through the Internet by any web-browser[11]. Among other activities, they can also see demonstrative measurements, control movement of displacement sensors, and control voltages of LEDs. Another remote laboratory, accessible by a web browser and available in seven languages, is described in Cmuk *et al.* 2008 [9].

From these examples, it can be noticed that in the teaching and learning process of measurement topics, tools are increasingly being used because of their expected benefits in higher education.

IV. THE TEACHING OF SOFTWARE MEASUREMENT

One of the relevant works related to the teaching of software measurement is the exploratory study conducted by Gresse von Wangenheim *et al.* 2009 [24]. This work was done to test the effectiveness of the game X-MED as an educational tool for complementing the lectures of a "Software Measurement Module". The game was aimed to the educational need of students at the master level and was tested among an experimental group. Fifteen Computer Science graduate students were part of that group. The duration of the module was 8 hours 45 minutes, including: pre-test, lectures, in-class exercises, the game and post-test. Only the students that were part of the experimental group played the game. X-MED presented a hypothetical scenario in which students have to develop a measurement program. In average the game lasts two hours. According to the Bloom's taxonomy, the highest level of learning expected for this module was "application". It is important to recall that this taxonomy has six levels of learning [20], including: knowledge, comprehension, application, analysis, synthesis and evaluation. It is important to remark that the exploratory study described above is the only one initiative that we identified in software measurement education, in which the learning objectives were explicitly indicated and considered according to the Bloom's taxonomy. It is also relevant that the learning effectiveness was investigated.

Due to the scarcity of publications related to educational approaches in software measurement, the authors of this paper have planned a series of research studies to have a better understanding of this issue. Among them, a literature review was performed to gain insights into how software measurement topics are taught in university courses, both in theory and in practice. The review consisted of a content analysis of publications appearing between 2000 and 2010, in which experiments with students were reported by university teachers. From the set of reviewed publications, we identified how software measurement topics are taught, specifically when students are exposed to practical measurement activities during the development of a toy or real project. Initial observations were reported in [25]. In addition, a survey was administered to teachers, students, practitioners, and consultants on software measurement in an academic environment and in software-related organizations. The objective of this exploratory study was to identify problems in the wording of questions and the structure of the questionnaires designed to answer the following two questions: How is software measurement addressed in undergraduate and graduate programs in universities? Do organizations consider that the graduating students they hire have an adequate knowledge of software measurement? The results of the exploratory study were reported in [26] and can be summarized as follows:

- Surveyed students and professors agreed that software engineering courses are usually mandatory in undergraduate and graduate programs, while software measurement courses are mostly optional.
- All surveyed teachers and students are of the opinion that software measurement is mainly taught through lectures, and more than 50% through case studies. In addition, in some courses, students are asked to develop toy or real projects in which size and total effort are collected.
- Topics, such as measures for the requirements phase and measurement techniques and tools, are receiving more attention in an academic context, and they confirm the findings reported in [25].
- Software measurement topics are mostly taught in graduate programs, although such courses are for the most part optional.
- The level of learning achieved by students is mostly assessed by written exams (75%) and projects (58%).

At the time this paper was written, a Web-based survey for university teachers and software measurement experts and practitioners was being conducted by using an enhanced version of the questionnaires previously mentioned. The survey for teachers, specifically considers questions related to: coverage of software measurement topics, expected level of learning per topic, types of teaching approaches used by teachers, ways for assessing students' learning; and type of skills developed for students in the measurement related courses. It is expected that the results of this web survey will provide a great source of information for researchers and educators in the software measurement field.

V. ASPECTS TO BE CONSIDERED IN THE TEACHING OF SOFTWARE MEASUREMENT

The previous sections have discussed two important and mature fields that are taught in engineering university programs, that is, math and measurement. Section III also presented the current trends in the teaching of these study fields with the underlying purpose of facilitating an active participation of students in their learning process. That these fields have reached maturity has enabled and facilitated researchers and educators to innovate and enhance the teaching of both subjects. In the case of the software measurement arena, however, this seems not to be the case.

Software measurement has not reached a mature level. This has been confirmed in the sense that no consensus has been achieved on most of the existing so-called 'software metrics' at the international context [27-29]. Moreover, the implementation and interpretation of software metrics with tools differ from each other [29].

In Jones 2008, there is a summary of 28 problems that need to be addressed in software measurement. One of them is the absence or inadequate training that undergraduate and graduate students receive at universities about this subject. Other problems mentioned by Jones are: the ambiguity in terminology, the lack of effective automation of software measurement tools and the ambiguity in the definition of metrics. These three problems may affect the characterization of the topics to be taught at universities. To tackle this issue, new educational approaches should put emphasis in the definition of "what to teach" and "how to teach". In the case of software measurement, the "what" has to do with the topics that have to be taught, for example: the basis of software measurement, the measurement process, techniques and tools, measures related to the requirement phase or other phases, etc. The "how" includes the educational approaches suitable for students to achieve the targeted learning outcomes.

In addition to the problems identified in Jones 2008, the literature points out that software measurement is considered as a difficult and time consuming task [30-33], which has become apparent in the classrooms. For example, in the exploratory study conducted by Gresse von Wangenheim *et al* 2009 (see section 4), they had difficulties in demonstrating a positive learning contribution to master level students by using the X-MED game. According to the authors, one of the explanations is that "the degree of complexity of the game requires a more comprehensive understanding of software measurement, project management and the CMMI framework" [24], p.26] that students could not gain in few hours lectures. Another study conducted by Buglione and Lavazza 2010 [34] also reported the difficulties experienced by undergraduate students in understanding the contribution of measures for controlling and monitoring projects. This was evidenced in students without professional experience who were using a procedure designed to help project managers choose proper project quantitative indicators. The previous discussion led us to identify aspects to be considered in the teaching of software measurement. They are as follows:

1) Prioritizing topics:

One of the constraints in education is the limited timeframe. This limitation poses a challenge to teachers to focus their attention and effort on important topics. This means "depth instead of coverage" [19]. Two ways of addressing relevant topics can be: spending more time on it or teaching for a higher level of understanding. The latter alternative is preferable because of the unavailability of time that is usually the case at university education. In addition to setting the priorities of topics, it is crucial to select the most suitable instructional and assessment methods in order to reach the learning goals. For achieving higher-order levels of cognition (i.e. analyze, evaluate and create), students have to learn progressively. This means, learning first the fundamentals of the study domain, and specialized topics later on. The resources assigned to each topic depend on the expected level of learning. For example, teaching "measurement concepts" might require fewer resources than teaching "how to measure the functional size of the software". In the first case, the recognition and explanation of measurement concepts is expected from the students, while in the second case the solution of sizing problems is expected by using hypothetical or real life software. Resources can include: the time spent by teachers and students in the classroom; preparing teaching material; doing assignments; and assessing students work.

2) Fostering deep understanding and higher order levels of learning in students

To the present, evidences show that the most common instructional approach in universities is expository teaching [19, 25, 26]. This approach, however, is not enough for students to reach a deep understanding and transfer their knowledge to new contexts outside the classrooms. Deep understanding can be achieved when students become engaged and motivated. One way of doing so is by allowing them to do hands-on activities and being exposed to problem-solving tasks and projects. As Biggs states "all students want to do something" [19]. This implies that the role of teachers becomes essential to students' motivation. Motivation and active participation of students are key aspects for learning as they allow them to reach a deep understanding of the topics being studied. They also facilitate to store all the acquired knowledge in their long term memory and retrieve it when needed in familiar and unfamiliar contexts [19, 20]. This way, students will be able to explain and analyze a given situation which, in turn, allows them to judge what is good and wrong and to propose new ways of doing something. A deep understanding is achieved when learners go through all cognitive levels, from memorizing concepts or terminology to applying their knowledge in solving new problems or creating new solutions. As software measurement is still an immature field, these abilities are crucial in software measurement. This means that future software engineering professionals will face continuous changes in the field and have to make decisions to deal with such changes.

3) Teaching in context

One of the common complaints of organizations in the software industry is that university students are not prepared to solve real life problems. One of the explanations for this situation is the existing gap between the academic programs and the industrial environment [25]. University students need to be prepared to confront and interact with real problems. One way to do it is by promoting the teaching and learning in context [21]. The underlying idea is that a person will learn provided that he/she uses knowledge in a context that presents the same challenges of the real world [35]. For example: a child will learn to swim by practicing in a swimming pool. Similarly, a software engineer will learn to measure the functional size by measuring real life software and analyzing requirements specifications documents. Amongst others, some ways to activate new experience can be: games, simulations, role playing, outdoor-based experiential instruction, work based learning and internships [19, 35].

4) Developing skills

Instilling in students the development of skills is as important as the acquisition of new knowledge. The learning of skills requires experience and practice [36]. Skill is “the capacity to perform a given type of task or activity with a given degree of effectiveness, efficiency, speed or other measure of quantity or quality” [36]. Skills can be classified as: 1) cognitive or intellectual; 2) motor or psychomotor; 3) personal or reactive; and 4) interactive or interpersonal [36]. The first one is associated with problem solving (familiar or unfamiliar), critical thinking, decision making, and all skills that demand the use and management of the mind. The second refers to physical action, perceptual acuity and other skills that use and manage the body. The third has to deal with attitudes and feelings, habits, self-discipline and self-control. And, the fourth involves social habits and the management of relationships and interactions with others. Another view to classify the skills is associated to their level of complexity. According to this perspective, all the aforementioned skills can be considered as reproductive or productive. Reproductive skills are related to the activities that are recurring or repetitive; while productive skills have to do with strategy or planning according to a specific situation that needs to be solved [36]. The students’ learning has to go beyond the development of reproductive skills to productive ones. In doing so, students, for example, have to be able to design and optimize an algorithm instead of only using it. Furthermore, they have to be capable to measure the functional size of different types of software instead of applying the rules of sizing methods in similar exercises. In addition, they will need to work in different teams during their university years in order to improve their interpersonal and persuasive skills instead of working with the same group of close friends.

5) Facilitating learning by effective assessment

Deep learning can be achieved by using two approaches: performance assessment and formative assessment [19, 20, 37, 38]. The former is used to assess functioning knowledge (i.e. how to do something) in its appropriate context (i.e. real-life professional problems). The latter, on the other hand, is characterized by the provision of feedback to students during their learning process. This means that students are informed about how well they are doing and what needs to be improved based on the results in the assigned tasks. Performance and formative assessments are complementary in the sense that they allow teachers and students to know, for example, how well a piece of software can be measured. It is important to remark, however, that real life problems might have more than one possible solution. Taking this into account, the assessment should consider the limitations that an educational environment may bring to the problem. If we refer to the previous example, the limitations could be: the time constraint for student to measure a piece of software and the amount of details provided in the SRS (software requirements specifications) document. This could happen when the students are somehow unfamiliar with the real problem. Some formats used for assessing functioning knowledge include: student presentations; poster presentations; individual and group projects; reflective journal; case study portfolio; and capstone projects [19]. Not only is the teacher’s assessment a useful mechanism but also peer–assessment. One example of a student presentation might include the analysis of three software measurement tools based on criteria. In this case, the student have to present the analysis to a jury composed by two teachers, three classmates and one software practitioner.

In summary, this paper maintains that math and measurement are two crucial subjects to be taught in software engineering education. In the specific case of software measurement, it is important to be clear about what and how to teach, and how to assess the students’ learning. In this respect, the intention of this article was to review relevant concepts and to provide examples to illustrate how these issues can be addressed to meet industry needs.

References

[1] C. Wohlin, "Are individual differences in software development performance possible to capture using a quantitative survey?," *Empirical Software Engineering,* vol. 9, pp. 211-28, 2004.

[2] S. K. Dey and M. A. Sobhan, "Guidelines for preparing standard software engineering curriculum: Bangladesh and global perspective," in *2007 10th International Conference on Computer and Information Technology (ICCIT 2007), 27-29 Dec. 2007*, Piscataway, NJ, USA, 2007, pp. 1-6.

[3] E. F. Barry and C. Norris, "Work in progress - project ClockIt: profiling and improving student software development practices," in *Frontiers in Education, 2005. FIE '05. Proceedings 35th Annual Conference*, 2005, pp. S1E-18.

[4] D. McFall, *et al.*, "Software processes and process improvement in Northern Ireland," in *Proceedings of the 16th International Conferences on Software & Systems Engineering and their Applications*, Paris, France, 2003.

[5] D. L. Parnas, "Software engineering programs are not computer science programs," *Software, IEEE,* vol. 16, pp. 19-30, 1999.

[6] C. Ghezzi and D. Mandrioli, "The Challenges of Software Engineering Education," in *Software Engineering Education in the Modern Age*. vol. 4309, P. Inverardi and M. Jazayeri, Eds.: Springer Berlin / Heidelberg, 2006, pp. 115-127.

[7] M. Hawthorne and D. Perry, "Software Engineering Education in the Era of Outsourcing, Distributed Development, and Open Source Software: Challenges and Opportunities," in *Software Engineering Education in the Modern Age*. vol. 4309, P. Inverardi and M. Jazayeri, Eds.: Springer Berlin / Heidelberg, 2006, pp. 166-185.

[8] H. van Vilet, "Reflections on software engineering education," *Software, IEEE,* vol. 23, pp. 55-61, 2006.

[9] D. Cmuk, *et al.*, "A Novel Approach to Remote Teaching: Multilanguage Magnetic Measurement Experiment," *Instrumentation and Measurement, IEEE Transactions on,* vol. 57, pp. 724-730, 2008.

[10] C. Eugène, "How to teach at the university level through an active learning approach? Consequences for teaching basic electrical measurements," *Measurement,* vol. 39, pp. 936-946, 2006.

[11] A. V. Szarka, ""Engineering is Fun" – in the instrumentation Laboratory," *Journal of Physics: Conference Series,* vol. 238, p. 012008, 2010.

[12] IEEE ACM. (2004, Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering. *SE2004 Volume – 8/23/2004,* 135. Available: http://sites.computer.org/ccse/SE2004Volume.pdf

[13] M. T. Battista. (1999, February 1999) The Mathematical Miseducation of America's Youth. *Phi Delta Kappan*. 424-433.

[14] D. Arnold. (2003). *Doing the Math and Making an Impact*. Available: http://www.ima.umn.edu/newsltrs/updates/summer03/

[15] W. T. Gowers. The importance of mathematics. Available: http://www.dpmms.cam.ac.uk/~wtg10/importance.pdf

[16] BIPM. (2003). Available: www.apmpweb.org/.../METROLOGY_MODULE_2003_11_20.ppt

[17] National Physical Laboratory (NPL). (2011). *Engineering Measurements*. Available: http://www.npl.co.uk/about/

[18] K. Kariya and S. Takayama. (2006, Measurement Science for Liberal Education to rise up Scientific and Technical Sense of Society. *Measurement science review Volume 6, Section 1(No. 4,),* 54-57. Available: http://www.measurement.sk/2006/S1/Kariya.pdf

[19] J. Biggs and C. Tang, *Teaching for quality learning at university*, Third ed. Buckingham: Society for Research into Higher Education & Open University Press, 2007.

[20] L. Anderson, *et al.*, *A taxonomy for learning, teaching and assessing. A revision of Bloom's taxonomy of Educational Objectives*. New York: Addison Wesley Longman, Inc, 2001.

[21] M. Crawford and M. Witte. (1999, November 1999) Strategies for Mathematics: Teaching in Context *Educational Leadership*. 34-38. Available: http://www.ascd.org/publications/educational-leadership/nov99/vol57/num03/Strategies-for-Mathematics@-Teaching-in-Context.aspx

[22] S. Larsen and M. Zandieh, "Proofs and refutations in the undergraduate mathematics classroom," *Educational Studies in Mathematics,* vol. 67, pp. 205-216, 2008.

[23] A. E. Castro Sotos, *et al.*, "Students' misconceptions of statistical inference: A review of the empirical evidence from research on statistics education," *Educational Research Review,* vol. 2, pp. 98-113, 2007.

[24] C. Gresse von Wangenheim, *et al.*, "Empirical evaluation of an educational game on software measurement," *Empirical Software Engineering,* vol. 14, pp. 418-452, 2009.

[25] M. Villavicencio and A. Abran, "Software Measurement in Software Engineering Education: A Comparative Analysis," in *International Conferences on Software Measurement IWSM/MetriKon/Mensura 2010*, Stuttgart, Germany, 2010, pp. 633-644.

[26] M. Villavicencio and A. Abran, "Facts and Perceptions Regarding Software Measurement in Education and in Practice: Preliminary Results," *Journal of Software Engineering and Applications* vol. 4, pp. 227-234, 2011-04-29 2011.

[27] A. Abran, *Software metrics and software metrology*. New Jersey: IEEE Computer Society / Wiley Partnership, 2010.

[28] C. Jones, *Applied Software Measurement: Global Analysis of Productivity and Quality,* Third ed.: McGraw-Hill Osborne Media, 2008.

[29] H. Yazbek. (2010, August 2010) Metrics Support in Industrial CASE Tools. *Software Measurement News: Journal of the Software Metrics Community*. 40.

[30] J. Iversen and O. Ngwenyama, "Problems in measuring effectiveness in software process improvement: A longitudinal study of organizational change at Danske Data," *International Journal of Information Management,* vol. 26, pp. 30-43, 2006.

[31] A. Gopal, *et al.*, "Measurement programs in software development: determinants of success," *IEEE Transactions on Software Engineering, ,* vol. 28, pp. 863-875, 2002.

[32] C. Gresse von Wangenheim, *et al.*, "Software Measurement for Small and Medium Enterprises - A Brazilian-German view on extending the GQM method," in *7th International conference on Empirical Assessment in Software Engineering (EASE), Keele University*, Staffordshire, UK, April 8 -10th, 2003.

[33] M. Diaz-Ley, *et al.*, "Implementing a software measurement program in small and medium enterprises: a suitable framework," *Software, IET,* vol. 2, pp. 417-436, 2008.

[34] L. Buglione and L. Lavazza, "Suggestions for improving measurement plans: a BMP application in Italy," in *IWSM/MetriKon/Mensura 2010*, Stuttgart, Germany, 2010, pp. 361-379.

[35] L. Lindsey and N. Berger, "Experiential approach to instruction," in *Instructional-Design Theories and Models*. vol. III, C. Reigeluth and A. Carr-Chellman, Eds., New York: Routledge, Taylor & Francis Group 2009, pp. 117-142.

[36] A. Romiszowski, "Fostering skill development outcomes," in *Instructional design theories and models*. vol. III, C. Reigeluth and A. Carr-Chellman, Eds., New York: Routledge, Taylor & Francis Group, 2009, pp. 199-224.

[37] Wikipedia. (2011). *Formative assessment*. Available: http://en.wikipedia.org/wiki/Formative_assessment

[38] J. Biggs, "Assessing for learning : some dimensions underlying new approaches to educational assessment," *Alberta journal of educational research,* vol. 41, pp. 1-17, 1995.