Issues in Student Valuing of Software Engineering Best Practices

Stephen T. Frezza, PhD, PSEM
Software Engineering
Gannon University
Erie, PA USA
Frezza001@gannon.edu

***Abstract*—This paper outlines the need for valuing of software engineering skills as a means to improve software engineering education. It presents a brief introduction to affective domain learning, and a survey of the education literature on software engineering skills related to software testing and quality assurance, which suggests that the competencies and skills needed extend beyond cognitive-domain learning. It then proposes a means for studying student valuing of these 'best practice' skill areas.**

Keywords—Affective Domain; Software Engineering; Best Practices

I. Educating for Engineering Best Practices

Engineering and computing education is driven more and more towards emphasizing the technical needs of graduates, while at the same time employers are faced with increasing reliance on their employees to innovate both their products and their processes. New software engineering graduates entering the workforce are often at the tip of the technology transfer cycle: not only do they need to be able to learn best practices before they join the workforce, they also need to be able to champion transferring these practices into their organizations. In a recent definition of software engineering competencies (SWECOM) defined it this way: "an additional competency of all software engineers is to instruct and mentor others, as appropriate, in the methods, tools, and techniques used to accomplish those activities." [1] This shift in engineering education is significant – Engineering classrooms are increasingly becoming schools of best-practice education.

In contrast with traditional science education emphasizing theory [2], effective engineering education emphasizes "practice applied with judgment" [3]. These engineering 'best practices' look at the size, types and necessary product development activities [1]. Typical core content in engineering education tends to focus on theory and/or small-scale component development. The difficulty is that "theoretical knowledge does not by itself lead to improved authentic problem solving skills" [4]. Second, this is usually far removed from the practitioner demands for technical 'crosscutting' skills expected of practitioner work.

While the best practices and essential competencies in any engineering domain may be identified [5], the difficulty is actually getting students to value these practices. Students may learn a particular best practice, demonstrate competence with it, and be able to describe when it should be used, but do not utilize that practice when not specifically required. In educational theory, this is termed a 'failure to transfer' knowledge and skills learned in one context to other (appropriate) situations [6]. This leaves them unwilling, and likely unable to champion the transfer of this particular practice into their workplace.

They type of learning described here is generally classified as *characterization* within affective domain learning. The affective domain describes learning objectives that emphasize a feeling tone, an emotion, or a degree of acceptance or rejection. *Characterization* by value or value set is to act consistently in accordance with the values he or she has internalized. Examples include: to revise, to require, to be rated high in the value, to avoid, to resist, to manage, to resolve. [7] In particular, the intent with teaching any engineering best practice is for students to select and attempt to use the best practice in appropriate situations without supervision.

The motivation for this work stems from research in how students come to know, and know engineering: Attitude plays a key role in patterns of knowing [8] and affects what students know and how they come to know it [9]. This also stems from author experience teaching different software engineering skill areas, including requirements management, software testing, software design and user-interface design. While this work will focus on a particular 'best practice' from software engineering, the hope is that the results of the final study would have some applicability for assessing student valuing of an identified best practice in other engineering specializations.

One of the difficulties in combing the literature on issues in teaching best practices is having a consistent nomenclature for classifying the practices or skill areas associated with 'best practice'. These skill areas both reflect the community definition of what is needed by engineering students in a particular discipline and outline the core content where 'best practice' learning is expected. The particular skill sets would be different for other engineering disciplines, but a taxonomy for a particular domain is necessary for the study. The mechanism selected here is to use expert-definitions for software engineering skill areas that both reflect teaching practice and practitioner needs [1].

II. Software Engineering Skill Areas

Recent work sponsored by the IEEE Computer Society provides an excellent starting point for defining skill areas in software engineering that are wanted by new graduates. This

work in software skill development presents a model for five life-cycle skill areas and nine crosscutting skills, each with identifiable sub-skills that map to practice and the expert literature in current software engineering practices. Table I lists these 14 different skill areas, which serves as a mechanism for grouping best practices as presented in the literature.

TABLE I. SWECOM SKILLS AREAS [1]

Life Cycle Skills	Crosscutting Skills
Software Requirements	Software Process and Life-Cycle
Software Design	Software System Engineering
Software Construction	Software Quality
Software Testing	Software Security
Software Sustainment	Software Safety
	Software Configuration Management (SCM)
	Software Measurement
	Human-Computer Interaction

The question, always germane to engineering educators, is what is the 'best practice' that should be taught in a particular software engineering skill area. Standard terminology for these skill areas is extremely helpful for collecting and classifying issues encountered in teaching students best practices. For example, issues in teaching model-based software testing [10,11] or test-first development [12] can both be grouped as issues under *Software Testing Skills.*

SWECOM similarly provides a useful lexicon for exploring best practices that can/should be included in software engineering education at different levels, as well as means to examine expectations of graduates in these skill areas. It defines five levels, Technician, Entry Level Practitioner, Practitioner, Technical Leader, and Senior Software Engineer, where the intent is that new computing graduates are at mostly at the 'Entry Level' [1]. The expectation is that these levels will play a role in disambiguating student cognitive and affective domain learning – and may be a potential contributor to issues with student characterization: the less they know, the less they may value the skill area.

III. TESTING AND QUALITY ASSURANCE SKILL AREAS

While there is extensive literature on methods for teaching the various software engineering skill areas, learning "practice applied with judgment" still appears to have issues. A short survey of the literature on teaching Software Testing and its related Software Quality skill areas yields assertions and conclusions like the following:

- Many IT professors teach content and then expect students to solve problems automatically without being shown the process involved. [13]
- Teamwork supporting tools and project roles are also neglected issues. [13]
- Attention not be paid to aspects related to code quality and use of design patterns for software architecture and resource planning by students on projects. [13]
- Projects given to the students did not match the realities experienced by industry programmers: larger systems, many features and much higher expectations of code quality. [14]
- The most common casualty is the code verification; typically, there are very few unit tests, there is no measurement of code testing coverage, and so forth. These projects fail to teach sound software engineering practices. [15]
- Teachers not always concerned about unit test implementation or user interface usability. [13]
- One of the biggest impediments of developing testing practices are the lack of knowledge in adopting new techniques. [16]

What these comments suggest is that teaching software quality, testing and related verification and validation techniques can be difficult. The point though, is not the difficulty, or the advantages of this or that particular method of teaching these skills, but rather a consideration of the broader sense that the expectations for the skillset(s) involved in just these related skill areas are broad, and not easily addressed. The question that this preliminary work attempts to identify is the various contributing expectations, and hypothesize why these expectations might remain unmet in many educations settings.

A recent detailed study of software testing practices and skill needs in industry expanded the technical skills to examine other professional competencies needed [16]. The results, both in examining the literature on software testing skills, interestingly ranked non-academic competencies and values significantly more often among those ranked as more important. These findings are summarized in Table II.

TABLE II. SOFTWARE TESTING COMPETENCIES [16]

Characteristic	Rank	Characteristic	Rank
Communication skills	7	Diplomacy	3
IT background	6	Need for challenge	3
Need for variety	6	Domain knowledge	3
Details oriented	5	Courage	3
Curious	4	Creative	2
Passion for quality	4	Proactive	2
Patient	4	Structured	2
Testing experience	4	Team player	2
Achievement orientated	3	Mindset to find bugs	2

Table II shows characteristics like "Details-oriented" and "Curious" or "Passion for quality" all listed above cognitive skills like "Testing experience." This finding (consitent with [17]) suggests that there is a significantly different set of learning goals involved in helping students become successful with software testing and/or quality assurance skill areas.

The question then, is what are these additional learning goals that should be included with these best practice skill areas. For example, passion for quality was noted as follows:

> Participants "expressed their desire to improve the quality in the products they are working on and taking pride in participating in the delivery of a high quality product. They talked about their joy in investigating and finding defects which will lead to a better product." [16]

One could view these 'desires' or 'taking pride' as inherent personality traits, but this argument ignores the literature

suggesting that attitude and affect are both learnable aspects of a topic [18] [19] [20]. Hence the need for edcuators to better model these types of affective domain learning outcomes, and be able to assess them reliably.

IV. WIP: Valuing Best Practices Approach

An example of an engineering best practice in software engineering comes from the software testing skill: the principle of test-first development (TFD). TFD is well suited to the small-scale development that most students engage in while at university and is required in multiple commercial agile development methods increasingly used in industry. Completely consistent with my teaching experience, research suggests "student attitudes towards test-first development may be the primary hindrance for new novices accepting and adhering to early test case development" [17].

Students' attitude of, and valuing of a software engineering skill or best practice (*e.g.*, test-first development) needs to be understood in order to assess, and ultimately improve their learning "practice applied with judgment." The result is that understanding the reciprocal nature of affect in learning appears to be especially important when teaching best practice methods [17]. Student affect, and particularly how they value a particular practice affects both their learning of that practice, and their ability to transfer knowledge and skill to their projects and workplace. If we as educators are to improve student learning of best practices, we have to be able to reliably assess their attitude toward, and valuing of the practices they are learning.

The project begins with the hypothesis that the fundamental obstacle to student retention of an engineering best practice is their internal valuing (characterization) of said practice. It focuses on assessing students' beliefs with respect to best practices. While we cannot assess students' inmost beliefs, we can identify what matters to them in practice by looking at what they actually do [21] [22]. This allows us to measure their behavior using three categories of the affective domain: receiving, responding and valuing [23] as steps for assessing their development toward organization and characterization. Computing best practices, and the value that these hold for students are both cognitive and affective in their structure. Consequently, this project entails developing and testing semantic differential [9] and multi-response checklists for assessing both cognitive and affective dimensions [24] [25] of student attitudes toward computing best practices. The core of this work will focus on the development and validation of questionnaires for examining students' values and structuring of values with respect to selected computing best practices. The longer-term goal is to establish valuation of software engineering practices in the context of Conceptual Change theory and methods [19] for the improvement of best-practice education in engineering.

A. Methodology Outline

One of the key shortcomings to assessing the affective and cognitive properties of attitudes is the need to address the interdependence between conceptualization and the measures used to assess that construct. The plan is to (1) Select a best practice and target audience (2) develop a set of assessment instruments for assessing student valuation of that practice, and then (3) to apply this instrument set to appropriate software engineering courses that explicitly include the particular 'best practice' identified by the research team.

Key to this initial study would be to identify a course/best practice with sufficient populations to enable some measure of statistical reliability. The expectation is to utilize the instrument set both in pre- and post- setups in an effort to establish a baseline on the effectiveness of current methods used in the target courses. The selection will be identified in a prior semester, in anticipation of pilot implementation in the following semester when the course is offered. With a core topic selected, then the work would be to identify the key terms and concepts that are involved with that best practice in order to develop draft semantic differential and multi-response checklist instruments around the cognitive and affective components of that particular best practice. The instrument set would be developed from other studies in science education [23,24]. Where a control group can be established, this also would be of use. The plan for this work in progress involves utilizing software testing best practice(s) for instrument development, as this is both common to many software engineering courses, and the work in cross-cutting skills [16] may also prove useful. However, the actual selection of the particular best practice area is subject to negotiation with the faculty of record, and may not be what is feasible for pilot deployment.

Semantic differential scales are used to assess dimensions of connotative feeling toward an object, person, or concept. These are most often assessed along one to three dimensions of meaning: strength, value and/or activity. These are represented as polar adjective pairs, with seven-valued selections indicating relative strength. For this study, a set of at least ten adjective pairs assessing value will be used, such as:

Worthwhile…	Worthless
Good…	Bad
Cheap…	Expensive

Multi-response checklists attempt to approximate interval levels, and are used to assess attitude toward a particular concept. Also known as Thurston scales, these are generated by developing equal-appearing intervals that form an attitude scale. This scale will via a three-step process. The first step is to develop response statements by eliciting a wide range of attitude statements toward the particular best practice. Then duplicate and irrelevant statements will be removed. A team of faculty judges will be asked to judges were asked to sort the statements into eleven stacks representing the entire range of attitudes from extremely unfavorable (1) to extremely unfavorable (11). If the judges cannot consistently rate an item on its favorability or show a high degree of variability in their judgments, the item will be eliminated.

With these steps in developing two sets of scales, the scales would then be piloted in a smaller setting available at my home institution. After review and adjustment, this would be attempted with a wider, larger population with similar best-practice learning goals. The results should provide both an individual and an aggregate assessment of student valuing of

the best practice selected; certainly not useful for grading, but for understanding how students judge a particular engineering practice they have been taught to use.

Like other engineering disciplines, there are multiple target software engineering best practices that are core to effective practitioner competency [1] which exhibit difficulties in transfer [12]. In order to best establish an effective study of student valuation, it is critical to simultaneously identify a best practice that anecdotally exhibits difficulty in transfer and courses in which these methods are prerequisites, taught and/or applied with sufficient students to indicate meaningful results. Consequently selection of the best practice is both important, and a bit difficult, especially with respect to the timing of this pilot study. While this paper focuses on the skill areas surrounding software testing and quality assurance (*e.g.,* TFD), the initial instruments and study may focus on another area where the student learning and participants are more available.

V. Conclusions and Next Steps

This work-in-progess has presented a rationale, and a short review of student valuation as a component of affective domain learning, and why the assessment of affective domain learning for software engineering best practices can positively affect their ability to better utilize their skills. It proposes a means for developing scales to use for studying what students do with skill-related tasks to assess their valuing and proposes developing and testing semantic differential instruments for this purpose.

References

[1] IEEE Computer Society, "Software Engineering Competency Model v 1.0," 2014.

[2] Stephen Frezza, David Nordquest, and Richard Moodey, "Knowledge-generation epistemology and the foundations of engineering," in *Proceedings of the Frontiers in Education Conference (FIE'13)*, Oaklahoma City, OK, 2013, pp. 818-824.

[3] Alan Cheville, "Defining Engineering Education," in *Proceedings of the 121st Annual Conference and Exposision*, Indianapolis, IN, 2014.

[4] F.V. Christiansen and C Rump, "Getting it right: conceptual development from student to experienced engineer," *European Journal of Engineering Education*, vol. 32, no. 4, pp. 467-479, August 2007, DOI: 10.1080/03043790701333063.

[5] Capers Jones, *Software Engineering Best Practices: Lessons Learned from Successful Projects in the Top Companies*. New York, USA: McGraw-Hill, 2010, ISBN 9780071621618.

[6] D N Perkins and Gavriel Salomon, "Teaching for Transfer," *Educational Leadership*, pp. 22-32, September 1988.

[7] Karin Kirk. What is the Affective Domain anyway? [Online]. http://serc.carleton.edu/NAGTWorkshops/affective/intro.html

[8] Stephen Frezza and David Nordquest, "Engineering Insight: The Philosophy of Bernard Lonergan Applied to Engineering," *Philosophical and Educational Perspectives in Engineering and Technological Literacy*, vol. 2, June 2015.

[9] Tibert Verhagen, Bart van den Hooff, and Selmar Meents, "Toward a Better Use of the Semantic Differential in IS Research: An Integrative Framework of Suggested Action," *Journal of the Association for Information Systems*, vol. 16, no. 2, pp. 108-143, February 2015.

[10] Stephen Frezza and Wayne Anderson, "Interactive Exercises to Support Effective Learning of UML Structural Modeling," in *Proceedings of the Frontiers in Education Conference (FIE'06)*, San Diego, CA, 2006.

[11] Stephen Frezza and Mei-Huei Tang, "Testing as a Mental Discipline: Practical Methods for Affecting Developer Behavior," in *Conference on Software Engineering Education and Training (CSEE&T '07)*, Dublin, IE, 2007.

[12] Kevin John Buffardi, "Modeling Student Software Testing Processes: Attitudes, Behaviors, Interventions, and Their Effects," Computer Science and Applications , Virginia Polytechnic Institute and State University , Blacksburg, VA, PhD Thesis 2014.

[13] Malgorzata, and Magdalena Borys Plechawska-Wojcik, "Methods and technologies for quality improving of student team software projects," in *Global Engineering Education Conference (EDUCON)*, Marrakesh, Morocco, 2012.

[14] Joseph Buchta, Maksym Petrenko, Denys Poshyvanyk, and Václav Rajlich, "Teaching Evolution of Open-Source Projects in Software Engineering Courses ," in *22nd IEEE International Conference on Software Maintenance (ICSM'06)*, Philadelpia, PA, 2006.

[15] Václav Rajlich, "Teaching Developer Skills in the First Software Engineering Course," in *Proceedings of the International Conference on Software Engineering*, San Francisco, CA, 2013, pp. 1109-1116.

[16] Anca Deak, "Understanding the influence of human factors on testing activities in software producing organizations," Department of Computer and Information Science Faculty of Information Technology, Mathematics and Electrical Engineering , Norwegian University of Science and Technology, Trondheim, PhD Thesis 2015.

[17] Kevin Buffardi and Stephen H. Edwards, "Exploring influences on student adherence to test-driven development," in *17th ACM annual conference on Innovation and technology in computer science education (ITiCSE '12)*, Canterbury, UK, 2013, pp. 105-110, DOI=10.1145/2325296.2325324.

[18] Ursula Fuller and Bob Keim, "Should we assess our students' attitudes?," in *Seventh Baltic Sea Conference on Computing Education Research*, Koli National Park, Finland, 2008, pp. 187-190.

[19] Ebru Kaya and Omer Geban, "The effect of conceptual change based instruction on students' attitudes toward chemistry," *Procedia Social and Behavioral Scie*, vol. 15, pp. 515-519, 2011.

[20] Daniel R. Lynch, Jeffrey S. Russell, Jeffrey C. Evans, and Kevin G. Sutterer, "Beyond the Cognitive: The Affective Domain, Values, and the Achievement of the Vision," *Journal of Professional Issues in Engineering Education and Practice*, pp. 47-56, January 2009.

[21] Maizam Aliasa, Tahira Anwar Lasharia, Zainal Abidin Akasahb, and Mohammad Jahaya Kesotc, "Translating theory into practice: integrating the affective and cognitive learning dimensions for effective instruction in engineering education," *European Journal of Engineering Education*, vol. 39, no. 2, pp. 212–232, 2014.

[22] Linda Vanasupa, Jonathan Stolk, and Roberta J. Herter, "The Four-Domain Development Diagram: A Guide for Holistic Design of Effective Learning Experiences for the Twenty-first Century Engineer," in *Journal of Engineering Education*, 2009, pp. 67-81.

[23] Ursula Fuller and Bob Keim, "Assessing Students' Practice of Professional Values," in *13th ACM annual conference on Innovation and technology in computer science education*, Madrid, Spain, 2008, pp. 88-92.

[24] S. L. Crites, Leandre R. Fabrigar, and Richard E. Petty, "Measuring the Affective and Cognitive Properties of Attitudes: Conceptual and Methodological Issues," *Personality and Social Psychology Bulletin*, vol. 20, no. 6, pp. 619-634, November 1994, SAGE Social Science Collections.

[25] William W. Cobern et al., "Active Learning in Science: An Experimental Study of the Efficacy of Two Contrasting Modes of Instruction," Western Michigan University, Kalamazoo, MI , Research Report 2011.