Reflective Essays in Software Engineering

Richard L. Upchurch
CIS Department
rupchurch@umassd.edu

Judith E. Sims-Knight
Psychology Department
jsimsknight@umassd.edu

University of Massachusetts Dartmouth
N. Dartmouth, MA 02747-2300

Abstract *- Software engineering education has evolved over the past ten years as understanding of the issues related to the practice of developing software systems has increased. A part of that evolution is an increased appreciation that learning software development requires more than participating in a design project. The design project provides a context in which the social and technical aspects of software engineering can become visible, but students often fail to learn the intended lessons. We, like other academics, believe that active reflection on experiences during these activities promotes the acquisition of more meaningful and persistent learning. We further believe that writing can and should play a critical role in promoting that reflective learning when the writing assignments require students to explore connections that arise during project activity. This will occur, however, only if the learning environment supports students in the construction and management of the writing activity, and supports faculty in providing the necessary feedback to students regarding their ideas. In this paper, we describe how our incorporation of writing activities in software project courses has evolved over the past five years, and a formative evaluation of our current efforts in software engineering.*

I. Introduction

Shaw [1] suggested, ten years ago, that as the complexity of software systems grows software engineering encounters new intellectual bottlenecks. Unfortunately, as Shaw also indicated, last generation bottlenecks are not eliminated. Software engineering education must respond to the ever-changing tapestry of software engineering practice. Part of that response has been to recognize that learning software development requires more than writing programs. The educational focus of software engineering in the last decade shifted toward the design project [2].

From an educational perspective the design project exposes students to a more authentic context for learning software development, including the experience of working in teams (e. g., [3], [4]). This approach, sometimes referred to as situated cognition, claims that learning outcomes are tightly coupled to the problem context, and can only be learned in that context. Such contexts also enhance student motivation.

Engaging in large-scale projects may indeed introduce students to the realities of the software engineering industry, but what do the students take away from the experience? The typical instructional focus is to provide the experiences with little regard to what is actually learned from those experiences (a criticism leveled at engineering design education in general by Dixon [5]).

The recognition that simply providing experience on a large project is not sufficient to ensure learning has led other software engineering educators to compromise on realism to permit more focus on the processes underlying the development activity [6][7]. For instance, Werth [6] encourages the use of laboratory time for students to practice team techniques. Other variations of the process-oriented approach focus on making explicit student roles and tasks [8]. We argued in [9][10][11] for the integration of software process concepts and practices in the software engineering curricula. There we contended that those software process activities that are incorporated into the educational program should be modeled after practices identified as essential to support good software development efforts, referred to as best practices [12].

II. The Role of Reflective Practice in Learning

The acquisition of expertise (a) takes a long time to develop, (b) takes intense, focused effort on the part of the individual, and (c) resists direct tuition. Ten years of exposure to a field appears to be a necessary, but not sufficient condition for the development of expertise. The number of years of experience is a weak predictor of performance. Practice alone is sufficient to develop a certain level of competence, but, for most people, will not result in true excellence. Deliberate practice, defined as practice in order to improve, has been suggested as the key to increasing competence. In many studies, the critical variable was not the amount of time engaged in the activity, rather it was practice with the intent of figuring out how to get better [13].

0-7803-5643-8/99/$10.00 © 1999 IEEE

What is being learned during deliberate practice? Certainly students are learning the domain-specific knowledge that provides experts with their efficient problem solving of familiar domains, but they are also learning metacognitive skills. A number of studies have demonstrated that students who are encouraged to reflect on what they are learning learn better both on declarative and procedural tasks (e. g., Pirolli and Recker [14]). It is also clear that only highly effective students use metacognitive processing; most students, even at the university level, do not [15][16]. Finally, simple interventions in which students are asked to process metacognitively have been found effective in improving problem solving and, in particular, transfer to new problems [15][16]. For example, Berardi-Coletta et al. [14] presented problem-solving tasks to college students. When students were asked to explain how they were deciding what to do next and how they knew that their choice was a good one before and after every move, they performed better and were more likely to solve a transfer task. In addition, the metacognitive process participants were much less likely to make negative personal evaluations. Comparable results have been found with training students to give self explanations both in learning textual material and in learning problem solving skills [16][17].

One of the implications of the Berardi-Coletta et al. study is that when students are learning effectively, they are less negative about themselves. The obverse of that is that students must believe that they can and will succeed through their efforts. Bandura [18] has recently reviewed much evidence that change requires a feeling of self-efficacy. For example, studies of the effectiveness of instruction in mathematics have shown that effective instructional techniques have their major impact through improving students' tendencies to persist and their feelings of self-efficacy.

Providing learning environments in which students will develop metacognitive processes therefore has four positive outcomes. They will learn the course material better. They will learn how to monitor and regulate their learning. They will learn how to change the way they work to become better learners. They will develop the positive attitudes toward themselves and toward persistent learning that will enable them to engage in the enormous amount of deliberate practice necessary to acquire expertise in their field and to engage in lifelong learning.

III. A Framework for Reflective Practice about Learning

The process approach requires that the practitioner first devise a plan for a project, then collect measurement data as the project is done, evaluate the effectiveness of the plan using the data collected, and, finally, reflect on how to improve. One technique we believed would be effective in helping students do this was the learning essay [19][20]. A learning essay is a written composition prepared by students in response to an instructional directive. The intent of the learning essay was to focus the students' attention on their current activity, or help them make the connection between their current activity and some other material. The writing activity could be as simple as documenting observations about their current software engineering process, or describing the manner in which they decided to perform a task. Whatever the content, it should require that the students think through their cognitive processes and make plans on how to improve that process, as suggested in the last section.

A. Initial Expectations

From this belief, we introduced learning essays into software engineering/computer science courses several years ago. In this paper we report our experiences over the course of six semesters in a total of eight courses. The first efforts asked students to provide their initial expectations for the course. We wanted to use this activity in the spirit of an advanced organizer, asking students to begin to deal with the purpose of a particular course in their program of study. The assignment was an unstructured activity without much guidance. The results were usually submitted via email to the course instructor. The prompt for the activity was:

> "Write your initial expectations of the course. From the course title this is about <topic>. You should consider comments you may have heard from others, you may have other insights. As you begin the semester record your thoughts and expectations."

Typically the responses, as seen in Table 1, were what one would expect. The remarks were very general and suggested that little thought preceded the writing activity. In 1998, the median number of words was 80.

The initial expectations were augmented with a set of surveys. The surveys were intended to capture information regarding how students developed programs for their courses, in other words how they worked. The surveys were implemented as HTML documents and the results collected in an electronic design notebook maintained for each student[9].

B. Postmortem

This approach seemed on target, but we felt there was insufficient feedback during the course of a semester for the students to recognize and articulate any differences. To alleviate this weakness we instituted project postmortems, as suggested in [21] as an industrial best practice, in a sophomore-level computer science course. The postmortem provided a repeated focus on how student engage the software development activity to complete the projects assigned. The goals of the postmortem activity were to:

- Provide an environment that supports openness and constructive criticism.
- Provide an environment that encourages improvement.
- Provide situations that use self-reflection.
- Provide an environment that encourages students to share lessons learned.
- Provide an environment that supports easy collection and archiving of information.

The students first completed a postmortem survey that asked them to revisit what happened during the project (early projects had a 1 or 2 week timeframe), and what specific tasks they performed to complete the project. Wherever possible we made use of close-ended questions students could answer quickly and unambiguously. It also included estimates of the time they spent in each part of the project and the importance they attached to it, so it provided data about how they worked. They were then asked to identify some of the activities that they found helpful (would repeat) and detrimental (would avoid) to their work habits. These two questions were open-ended so students would have to generate answers.

S1: Since this course is about Software Engineering, I expect to become better at engineering software. I expect to learn more about the software process and the necessary steps that must be taken to write code that has quality and efficiency.
S2: I am looking forward to this course. I have heard that it is very tough, but also very rewarding. I hope to learn the process used to develop quality software.
S3: I am looking forward to the class. It appears to be an interesting class. It will be different from the 'just' programming aspect. The project will most likely be difficult but it will be interesting to see the final result.

Table 1: Students' Initial Expectations

What is does software engineering seem to be about? From the first two lectures and the first chapter of X, you should have gained some ideas about the issues software engineering treats.

How do the issues addressed in software engineering merge with your career goals? This, of course, isn't necessarily perfect. You don't have complete information regarding where you go from here. You do however have some ideas about how industry works and what benefits certain types of learning may have in the world-of-work.

As a student and potential professional, what is it about software engineering that appears important? This should be examined from the perspective of "what should you strive to achieve in this course." You should address the issue of what seems to be the primary learning goals for you and why these are identified as important. This should be phrased in terms of who you are now versus who you think you should be by the time you graduate.

Table 2: Scaffolding for Initial Expectations

Students also had collected measurement data on the project. During the first trial we asked students to collect size data on the source code developed for the project. The measurement template included the definitions the students

required for SLOC and documentation, specifying what was included and excluded from each. In subsequent projects data collection included effort estimates also.

The concluding activity of the postmortem was goal setting. The students were asked to establish a goal for the next project and identify activities they thought would help them achieve their goal.

C. Team reviews

From the essays in the first course, it was clear that students need as much practice as they can get in evaluating products and process. Thus, in addition to setting up the framework for their process improvement loop and revising the prompts and scaffolding, we also changed the nature of team reviews from informal and orally delivered to more formal, written documents [11]. Though an improvement, considerable work needs to be done to properly formulate this activity for students, thereby increasing its instructional effectiveness.

D. Culminating Activity

The survey, given at the beginning of the course, was given again during the last week of the semester. Repeating the survey was part of a culminating activity. This activity required an essay in which students compared their development behaviors prior to the course with those at the end. This comparison was submitted during the last week of the semester (see [9] for a more complete report of this activity). In the software engineering class, these activities were subsumed as part of the legacy document required of each project team. The culminating essay from guided by the prompt: "Provide a narrative from each team member that summarizes the experience(s) of the semester." Anecdotes from this activity are provided in Table 4.

Prompt:
What interests me from your responses is how will you know you achieved your goals? One could argue that successfully completing the course is one method of assessment. Receiving a high letter grade is some measure of success I suppose. Unfortunately that is an external measure that may or may not have anything to do with your goal. The way I phrased the statement was in terms of change. With that phrasing it seems valid to proceed by asking, what behavioral changes would you expect? These should be observable, and/or measurable. How will you know if you have reached these objectives?

Scaffold:
What differences in the way you approach software development should result in your study of software engineering and participation in the project portion of the course?
How will these differences manifest themselves in your behavior? Will these behavioral differences be evident to others that know you?
What evidence would you suggest to convince me the changes in your development approach are better?

Table 3: Expectations Follow-up

IV. The Learning Essay

We felt that reflective writing is a critical component of the software engineering classes. Unfortunately, our efforts over the years were only partially successful, as documented in the last section. The expectations activity, pre/post, did not elicit the depth of reasoning we expected or wanted. However, the results from the postmortem activity were encouraging. In reviewing the manner in which we requested the expectations, we realized we were getting precisely what we asked for--nothing. While we expected students to take the challenge of thinking through the ins and outs of courses with its rationale, there was really no practical reason why they should arrive at the outcome we wanted. This is consistent with the findings in cognitive and instructional research that students do not automatically or naturally think reflectively [15][22]. We wanted students to make connections, yet we were asking them to determine what is connected

and how with very little assistance. Hence, we revised the activity to provide metacognitive scaffolding [23][24][25]. In our case, we devised prompts (see Table 2) to direct students to think about certain issues in relation to the course and to their program of study. The results indicate that students provided more information about their expectations. In the Spring of 1999 the median number of words produced was 215.

We followed the initial expectations assignment with a second writing assignment, as detailed in Table 3. The scaffolding for the writing is given by a series of questions under the prompt. These scaffolds give the students ways to think about the writing task. In doing so these scaffolds help structure the response. The median number of words in response to this assignment was 288. Apparently the restructuring of the activity to include scaffolding works. The combination of the two writing assignments had the students dealing with the issues intended. Additionally, the students' responses to the first writing assignment guided the choice of the second, and suggested the scaffolding.

Scaffolding was also added to the culminating essay. In the legacy document required in software engineering each team member was required to complete a short essay on technical lessons and managerial lessons learned. These lessons learned sections were given with prompts (e.g., Each team member should write a short narrative regarding what management lessons were learned during the project.). Hence, for the software engineering students, the final essay was informed by:

- Initial expectations
- Initial survey
- Terminal survey
- Technical lessons learned
- Managerial lessons learned

S1: I found that I needed to be more diplomatic in my approach in dealing with team members. To work well as a team, all members must be open to suggestions, and be able to change portions of what they are doing, if the team feel this is best. I also found that there is a great skill involved in explaining to team members about products I developed. This, I feel is one of the most important lessons that I learned. If I produce something that I feel is what was required and is the best possible product, I must be able to explain my reasoning, firmly and coherently to a group, and be willing to reassess if necessary.

S2: While I consider myself a conscientious developer, using modeling techniques and planning, I was lacking in the area of team coordination. I still think I am lacking somewhat in that area, but I feel more prepared to encounter the situations associated with being in a development team. I feel that a course like this one can provide a preview to that, but what we experienced in our project was quite unlike a real world software project in many ways. But the ways in which it has improved my approach has made my time here worthwhile.

S3: I can apply the fundamentals of software engineering to any process; ensure complete and accurate requirements, choose a correct design based on the problem, and create a quality program that continually refines the process and leads to a better product.

Table 4: Culminating Essay Anecdotes

V. Conclusion

"The greatest danger to software engineering education curriculum designers is lack of imagination. If we are too narrow, too shortsighted, or too low in our aspirations, we will deprive the field of the skills it needs to satisfy society's requirements for broader scope and larger scale in computer-based systems."[1, p. 10]

To be able to apply the software process approach, students need both to understand how the process works and to be able to do it. Simply experiencing the process approach does not ensure that students can do it, particularly when they must plan their process and reflect upon the measurement data they have collected to write an improvement plan. Being required to write a learning essay before and after every development effort provides the basis and practice necessary for improvement. Such assignments alone do not promote improvement, because inexperienced students have inadequate metacognitive process skills to succeed. With elaborated prompts and scaffolding, however, students begin to ask themselves the right questions and to monitor their own understanding. This is the type of activity we believe prepares students with the skills in response to Shaw.

The learning essays that we have implemented provide a framework by which students can create their own process improvement loop, but we have just begun the process of creating a learning environment that facilitates the acquisition of metacognitive processes and the use of the process improvement loop. In software engineering, the students appeared to accept the tasks willingly. The remarks seemed to indicate that the students addressed the issues thoughtfully, though a thorough analysis is required. It was evident, though anecdotal, that the students were making the types of connections we were seeking. The initial expectations and follow-up essay will be revised or followed by an essay that requires students to create a plan of how they will proceed in their first development project.

Our results support several conclusions regarding postmortems (see [9] for a more complete discussion). First, students made gains regarding their development activity. Second, they saw the impact of their practices, Third, they began to connect their practices with potential improvement strategies. Finally, they could articulate the influence between the way they worked with effort and quality. Such activities helped students develop a broader, more accurate understanding of a, and their, software development process, and permitted them to begin to develop the process skills they need as professionals.

The project postmortems need to be restructured to ask students a series of questions to help them evaluate the measurement data more thoughtfully, i.e., scaffolded. This restructuring will then require them to review their previous plan in light of their evaluation and to write a revised plan for the next project. The systematic reviews of designs and programs will continue and we will experiment with evaluation of the reviews themselves to make that process more reflective.

The course postmortem will take the expectations documents and their postmortems as data to evaluate and write an improvement plan for their next course as well as an evaluation of how the course helped or failed to help them improve their processes. The latter will, of course, provide the data we need for our curricular continuous improvement loop.

VI. References

1. Shaw, M. "Education for the Future of Software Engineering," SEI, Carnegie-Mellon University, SEI-86-TM-5, 1986.
2. Tomayko, J. "Teaching a Project-Intensive Introduction to Software Engineering," SEI, Carnegie-Mellon Univeristy, 1987.
3. Denning, P., Menasce, D., & Gerstner, J. "Re-engineering the Engineering School," ASEE Conference Proceedings, 1995.
4. Moore, M. & Potts, C. "Learning by Doing: Goals and Experiences of Two Software Engineering Project Courses," in J. L.Dìaz-Herrera (ed.), *Software Engineering Education: 7th SEI CSEE Conference*. New York: Springer-Verlag. 1994, p. 151-164.
5. Dixon, J. R. "The State of Education," *Mechanical Engineering*, February 1991, pp. 64-67.
6. Werth, L. "An Adventure in Software Process Improvement," In J. L.Dìaz-Herrera (ed.), *Software Engineering Education: 7th SEI CSEE Conference*. New York: Springer-Verlag. 1994, p. 191-210.
7. Robillard, P., Mayrand, J. & Drouin, J. "Process Self-Assessment in an Educational Context," in J. L.Dìaz-Herrera (ed.), *Software Engineering Education: 7th SEI CSEE Conference*. New York: Springer-Verlag. 1994, p. 211-225.
8. Werth, L. "Software Process Improvement for Student Projects," IEEE 1995 Frontiers in Education Conference, 1995.
9. Upchurch, R., & Sims-Knight, J. E. "Integrating Software Process in Computer Science Curriculum," Frontiers in Education Conference, Pittsburgh, PA, November 5-8, 1997.
10. Upchurch, R., & Sims-Knight, J. E. "Designing Process-Based Software Curriculum," Proceedings of the Tenth Conference on Software Education and Training, Virginia Beach, VA, April 13-16, 1997. Los Alamitos: IEEE Computer Society Press, pp. 28-38.

11. Upchurch, R., & Sims-Knight, J. E. “In Support of Student Process Improvement,” Proceedings of CSEE&T'98, Atlanta, Georgia, February 22-25, 1998.
12. Brown, N. “Industrial-Strength Management Strategies,” *IEEE Software*, July 1996, pp. 94-103.
13. Ericsson, K. A. (ed.) *The Road to Excellence*. Mahwah, NJ: Lawrence Erlbaum, 1996
14. Pirolli, P. & Recker, M. “Learning Strategies and Transfer in the Domain of Programming,” *Cognition and Instruction*, 12, 1994.
15. Berardi-Colletta, B., Buyer, L. S., Dominowski, R. L., & Rellinger, E. R. “Metacognition and problem solving: A process-oriented approach,” *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 21, 1995, pp. 205-221.
16. Chi, M., de Leeuw, N., Chiu, M., & LaVancher, C. “Eliciting self-explanations improves understanding,” *Cognitive Science*, 18, 1994, pp. 439-477.
17. Bielaczyc, K., Pirollli, P. & Brown, A. "Training in Self-Explanation and Self-Regulation Strategies: Investigating the Effects of Knowledge Acquisition
24. Guzdial, M. & Kehoe, C. "Apprenticeship-Based Learning Environments: A Principled Approach to Providing Software-Realized Scaffolding through Hypermedia," *J. of Educational Multimedia and Hypermedia*, 1998.
25. Guzdial, M. "Software-Realized Scaffolding to Facilitate Programming for Science Learning," *Interactive Learning Environments*, 4(1), 1994, pp. 1-44.

Activities on Problem Solving," *Cognition and Instruction*, 13, 1995, pp. 221-252.

18. Bandura, A. *Self-efficacy: The exercise of control.* New York: Freeman, 1997.
19. Turns, J., Newstetter, W., Allen, J. & Mistree, F. "Learning Essays and the Reflective Learner: Supporting Reflection in Engineering Design Education," ASEE, 1997.
20. Turns, J. "Learning Essays and the Reflective Learner: Supporting Assessment in Engineering Design Education," Frontiers in Education Conference, Pittsburgh, PA , 1997.
21. Collier, B., DeMarco, T. & Fearey, P. “A Defined Process for Project Postmortem Review,” *IEEE Software*, July 1996, pp. 65-71.
22. VanLehn, K., Jones, R. M., & Chi, M. T. H. “A model of the self-explanation effect,” *Journal of the Learning Sciences, 2*, 1992, pp. 1-59.
23. Hübscher, R., Puntambekar, S. & Guzdial, M. "A Scaffolded Learning Environment Supporting Learning and Design Activities," Presented at AERA, Chigcago, IL, March 24-28, 1997.