Teaching Statement

Irwin Kwan

Based on my experiences learning and teaching software engineering, I have found that many engineering students are actively focused on *doing.* These students want to dive right into the technology without having to learn about theories and processes. As software engineering is fundamentally about planning and coordinating a complicated project, how can I make these students interested in planning software projects and following processes? To address the problem, I plan to structure my teaching around a *minimalist learning* philosophy, in which learners, such as students, learn information just-in-time in the context of their goals [2]. Minimalist learning theory has three main principles: to (1) allow learners to start immediately on meaningful tasks, to (2) minimize the amount of passive forms of training by allowing users to fill in the gaps themselves, and to (3) include error recognition and recovery activities in the instruction. Though minimalist learning theory was initially developed as a form of self-instruction, I incorporate these three principles into the classroom to ensure that learners are not only able to learn from me and their peers, but to also discover for themselves.

Learning from meaningful tasks with client projects

Students often see software engineering processes are often seen as unwieldy and cumbersome, but many students have not yet had the opportunity to work with a team of software developers who are trying to develop a product with vague requirements.

An effective way to give students a personal stake in the course is to involve them as stakeholders of another student team. The team has two roles: 1) A developer team that implements the system of a client team, and 2) a client team that provides requirements and approvals to a developer team. The client teams should propose a system that fulfills their personal needs, so they have a vested interest in its proper creation. This two-sided experience enables student teams to be on both sides of development and gain an appreciation of the communication needs of both sides. I experienced this team setup when I was a teaching assistant for requirements engineering at the University of Victoria, and received positive feedback about this arrangement. Teamwork enables students to cope with issues that they would otherwise not face as individuals, such as collaboration. Interacting with another client team encourages students to discover interpersonal skills, negotiation, and compromise.

Reducing passive learning using student-active breaks

I enjoy encouraging active forms of learning and discovery and facilitate this in my teaching by applying two techniques: an active-learning technique called student-active breaks [5] and increasing the visual materials that I use in class to support multiple learning styles [3].

Student-active breaks are periodic pauses in lecture in which students are asked to interact with the material for a short, controlled period of time. Students are asked to stop and discuss the material with others; sample activities include "pair and compare", where students can pair up with their neighbour and compare lecture notes to fill-in gaps; free-recall, where students write down the most important points from the lecture; or group and discuss, where students group together to discuss material and present it to the rest of the class. I have used some of these techniques in my open-source class and students responded favourably to the activities.

Students are known to recall what they both see and hear more effectively than seeing or hearing individually [6]. As a consequence I have been adopting a multi-sensory approach to teaching, in which I am incorporating more visual and animated elements to explain concepts. I have been adapting my presentation style in the past three years to move away from "bullet points" and students have responded well to these changes.

Allowing error recognition and recovery activities as part of instruction

Students are often uncertain about the effects of their actions on their marks and often do not perceive errors in their work until it's too late. To mitigate this risk, I incorporate multiple opportunities for feedback to student teams. This reinforces two objectives. First, it provides students the opportunity to recover from errors before it is too late, and second, it reinforces that in practice, a deliverable that is handed in is not a final product to be forgotten about.

In software engineering, I will reinforce *iterative development*, in which developer teams iterative over a working product in short process cycles. However, at the same time, I will encourage teams to engage in requirements planning and prioritization as well to ensure that they have a long-term outlook regarding their projects.

The student developer teams will present elements of their plans and designs frequently to both the instructor and their client teams. The instructor will ensure that technical elements are sound, and that proper processes are being followed while the client teams will ensure that the developer teams are implementing their requirements. Often, developer teams tend to promise more than they can deliver—this leads to a mid-cycle negotiation of what features to implement, which is an important part of software project management. Meeting frequently with stakeholders who provide valuable feedback enables the teams to reflect and correct their project's management and implementation accordingly.

Encouraging Learning and Discovery

In addition to making learning more applied to students' tasks and activities, I strive to create an environment in which students feel comfortable to interact with me. At the beginning of each lecture, I ask that students give a brief update to the class about lessons learned, challenges, and insights that their team has learned since the previous lecture. This encourages students to reflect on their project and their practices as they are doing them. In addition, I can draw upon my experiences with software development teams and reinforce experiences and problems the students experience with concrete, real-world examples.

I will also strive for diversity in the classroom. I am involved in efforts to increase diversity and involvement from minorities in engineering from my research in gender-oriented human-computer interaction and end-user programming (e.g. [4, 1]). I will incorporate learning resources from organizations such as the National Center for the Women in Information Technology (*ncwit.com*) and Disability Services into my teaching to ensure that every student in my class has the opportunity to learn, and that these students can learn not only from me, but from each other.

References

[1] Jill Cao, Irwin Kwan, Rachel White, Scott D. Fleming, Margaret Burnett, and Christopher Scaffidi. From barriers to learning in the idea garden. In *IEEE Symposium on Visual Languages and Human-centric Computing 2012*, pages 59–66, Innsbruck, Austria, 2012.

[2] John M. Carroll. *The Nurnberg Funnel: Designing Minimalist Instruction for Practical Computer Skill.* The MIT Press, June 1990.

[3] Neil D. Fleming and Colleen Mills. Not another inventory, rather a catalyst for reflection. *To Improve the Academy*, 11:137–155, 1992.

[4] Valentina Grigoreanu, Margaret Burnett, Susan Wiedenbeck, Jill Cao, Kyle Rector, and Irwin Kwan. End-user debugging strategies: A sensemaking perspective. *ACM Transactions on Computer-Human Interaction*, 19(1):5:1–5:28, May 2012.

[5] Kathy L. Ruhl, Charles A. Hughes, and Patrick J. Schloss. Using the pause procedure to enhance lecture recall. *Teacher Education and Special Education: The Journal of the Teacher Education Division of the Council for Exceptional Children*, 10(1):14–18, January 1987.

[6] Donald R. Woods. Developing students' problem-solving skills. *Journal of College Science Teaching*, 19(3):176–79, 1990.