

## **AC 2007-2868: AN ANALYSIS OF MULTI-YEAR STUDENT QUESTIONNAIRE DATA FROM A SOFTWARE ENGINEERING COURSE**

### **Valentin Razmov, University of Washington**

Valentin Razmov is an avid teacher, interested in methods to assess and improve the effectiveness of teaching and learning. He is a Ph.D. candidate in Computer Science and Engineering at the University of Washington (Seattle), expected to graduate in 2007. Valentin received his M.Sc. in Computer Science from UW in 2001 and, prior to that, a B.Sc. with honors in Computer Science from Sofia University (Bulgaria) in 1998.

# An Analysis of Multi-Year Student Questionnaire Data from a Software Engineering Course

## 1. Introduction

Improving student learning has been a long-standing goal of educators across all disciplines. To improve effectively and methodically, one needs to know what works well (and needs to be sustained) and what does not work well (and may benefit from changing). In a classroom environment, the two direct stakeholders, instructors and students, can both provide valuable perspectives on how things are going.

This paper presents an analysis of an extensive set of feedback data provided by students across 8 academic terms for an undergraduate introductory course in software engineering, taught at a large US public university. The feedback was gathered via end-of-term course-specific questionnaires, separate from and much more detailed than the typical university-sponsored course evaluations. In total, 162 students gave feedback, while 5 different instructors were involved with the course, one of whom – the author of this paper – was actively engaged in all 8 offerings.

To give the reader a sense of scale, the end-of-term student questionnaires featured 60-150 questions – mostly multiple choice questions, as well as some free-form short-answer questions. The subject of the questions were the course structure, the instructors' teaching approach, class sessions, readings, writing assignments, project experiences, tools, the feedback that students received from instructors and peers, as well as questions aimed at capturing student perceptions of what had worked well and what had not.

Among the encouraging results are that students almost unanimously report feeling better prepared for industry careers after taking the course. They also increasingly come out with a heightened appreciation for the value of incremental project development and of many of the “softer” (non-technical, human) issues in engineering. In contrast, the main aspects that our analysis identifies as needing further improvement are the choice of course readings, as well as a stronger emphasis on quality assurance practices and techniques for dealing with ambiguity – both aspects that students tend to find unfamiliar and unnatural. We also share a few surprises found in the data.

Our main contributions are the analysis of the rich body of collected data, as well as distilling groups of questions that have yielded particularly useful results, and categorizing those by target outcome: questions for evolving the course, for “reading” students' moods, and for getting students to reflect on their experiences. Many of these questions may be broadly applicable.

The remainder of the paper is structured as follows. Section 2 elaborates on relevant aspects of the course structure and describes our mechanism for collecting feedback data. Section 3 discusses what we have learned from our data analysis – first about the course, and then about the process of doing student surveys. We conclude in Section 4. To give the reader a concrete view into the nature of our questionnaires, the Appendix contains the full list of questions from the most recent end-of-term student questionnaire.

## 2. Course Structure and Context around Doing Student Questionnaires

The course is intended to be an introduction to software engineering for junior and senior students of Computer Science and Computer Engineering. Our main goal in teaching the subject is to enable students to learn how to work effectively in teams and to deliver value on long-term projects. Toward that end and in the spirit of experiential learning, students work on term-long projects in teams of 5-8 (typically), following the incremental delivery approach<sup>5</sup> with a short iteration cycle – we set intermediate project deliveries roughly once every 2.5 weeks. After each delivery, instructors facilitate in-class retrospectives. Instructors also meet separately with each project team to provide feedback, address questions, and “take the pulse” of the team. 2-3 times during the term, usually shortly after some of the project deliveries, students complete anonymous peer evaluations for their teammates, offering constructive feedback. Each student also completes 2-3 individual reflective writing assignments during the term, to which instructors provide extensive written feedback and follow-up questions, engaging students in personalized discussions. Lastly, there is a take-home final exam and (in some offerings) an in-class midterm.

Our experiences come from 8 terms over 4 years when we were involved in teaching the course. About 210 students took the course during that period; enrollments varied between 8 and 43 (the lower numbers coming from summer-term offerings), with the median at 27. Over this 4-year period the course has evolved in large part owing to valuable feedback from several sources, chief among which have been the extensive course-specific end-of-term student questionnaires that we designed, as well as post-course formal retrospectives by instructors.

The questionnaires were conducted either anonymously (unnamed, on paper) or quasi-anonymously (electronically, though instructors had promised not to look at the content until after final course grades were turned in). In either case, the act of completing a questionnaire, but not its actual content, counted for a small percentage of a student’s course grade (except in one offering when the lead instructor made it voluntary), which explains the high response rate. Given that there always was a mix of opinions – including criticisms we had not encountered during the terms – we infer that students did not feel overly pressured to say things we may like.

Post-course retrospectives (or post-mortems) based on all collected materials were done after 6 of the 8 course offerings by the respective instructors. These formal retrospectives took roughly 2 hours to complete and followed the familiar sustain/improve structure<sup>4</sup> – designating aspects to be sustained and identifying issues in need of improvement.

As a result of continually reexamining the inputs from questionnaires and retrospectives, the course structure has gradually evolved, though the main principles have remained unchanged:

- emphasis on both the technical and human aspects in software project development;
- students working on relatively large (term-long) projects in large teams;
- teams regularly presenting their latest project releases to both peers and guest experts;
- students being in charge of managing their own teams;
- students practicing reflective skills through writing assignments;
- extensive feedback from instructors to students (on projects and writings), and from students to instructors (on the course).

### 3. Interpreting the Data

In this section, we discuss our findings from examining the collected data. These are grouped into lessons we learned about the course, including what worked well, what did not, and what surprised us, as well as meta-lessons we learned about the process of gathering and analyzing data.

To ground our conclusions, we quote representative student responses, and in the case of multiple-choice questions we provide a series of class-average ratings for the aspect in question, tracked over several terms. Each multiple-choice rating (and hence each class-average rating for a given term) is on a scale of -2 (i.e., a strongly negative value of the corresponding aspect to the respondent) to +2 (i.e., a strongly positive value of the aspect in question). To offer a simple baseline of comparison, class-average ratings above +1.5 are rare (in the top 10%-15% of all responses), so we interpret the corresponding aspects to have been particularly successful during the term for which these high ratings are quoted; toward the other end of the spectrum, class-average ratings near or below 0.0 naturally indicate the presence of a serious problem in need of attention. As a third data point of reference, class-average ratings in the middle of that range – between 0.6 and 0.8 – are often comparable to what the default averaged response would be, corresponding to a case when the majority of the students do not have a strong opinion (either positive or negative) about the subject in question.

We caution that our dataset does not come from a controlled study – we simply obtained responses from as high a percentage of our students as we could in hopes of a broad representation. Hence, the suggested conclusions may contain biases. In practice, however, acting on such conclusions in the past has uniformly resulted in improvements of subsequent ratings on the corresponding aspects.

#### 3.1. What We Learned about the Course

Many aspects of the *course structure* have been fine-tuned over the years. Students and instructors have liked most of the results. Next, we highlight the components that worked very well, then present the ones that did not work so well, and finish with a few that raised eyebrows.

##### 3.1.1. What Worked Well

Students appreciated large *team sizes* (but not too large) that allowed them to work with several peers while learning how to do effective group work on non-trivial projects. The class averages for this aspect were 1.55, 1.40, 1.62, and 1.38 for the terms when the team sizes were in the range 5-8, and 0.43, 0.12, 1.00, and 0.29 for the terms when team sizes were either bigger or smaller (Figure 1(a)). In a free-form response, one student noted, “The experience in a large group project was great,” echoing the feelings of many others.

Another much appreciated aspect was the *incremental delivery approach*. Class averages were 1.50, 1.46, 1.45, 1.62, 1.50, and 1.86 for the terms when we required it, and only 1.09 and 1.00 for the (very first two) terms when we purely recommended it (Figure 1(b)).

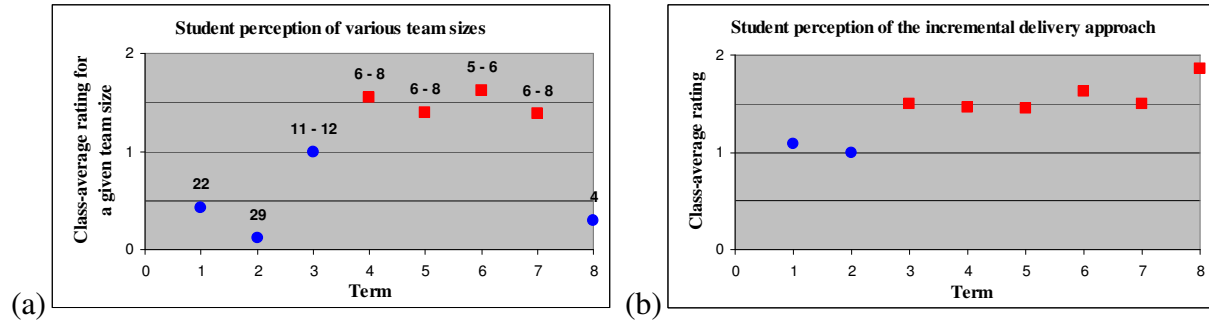


Figure 1. (a) Student perception of team sizes. Each dot in the graph is labeled with the range of team sizes for the respective term. Most welcome were teams of size 5-8 (corresponding to the red square dots), while smaller or larger teams (corresponding to the blue circle dots) received less favorable ratings in the respective terms when they occurred.

(b) Student perception of the incremental delivery approach. In the first two terms the approach was recommended, while subsequently its use was required. In the graph, the lower early ratings are denoted by blue circle dots, whereas the higher later ratings are marked by red square dots.

Students also felt generally positive toward *tools and environments facilitating group work and large-scale development* – such as CVS, Eclipse / Visual Studio, wiki web pages, and project mailing lists – but only to the extent that those were noticeably helping the development on their specific projects. The corresponding class-average ratings were as high as 1.50 in some terms, and as low as 0.62 in other terms – depending on how seamlessly the tools were incorporated into the development process of the teams (which often depended on the choice of projects too). This wide swing of attitudes between terms and teams may have a plausible explanation in the lack of preliminary positive experiences<sup>6</sup> of some students with the particular tool in question, thereby making the tool feel more like a hindrance and less like an aid.

*Guest speakers* – typically technical managers or lead engineers from industry – were popular, adding their enthusiasm and refreshing perspectives to the classroom environment. Several students wrote that the guest lectures were among the “most memorable” for them. Instructors also loved the guests’ impact in corroborating, as industry experts, ideas regarding specific techniques and tools that had been previously suggested in class. The class-average ratings for guests ranged between 1.27 and 1.55 for 6 of the 8 terms (5 of those being the most recent terms), with the ratings for the other two terms being markedly lower. We believe the difference can be explained by the fact that some of the guests in those two terms offered more controversial points of view.

The *use of technology to facilitate in-class activities and help engage shy students* received positive responses from many. We deployed a popular Tablet PC-based classroom interaction system called Classroom Presenter<sup>1</sup>, offering students an additional mechanism to interact with the instructor – by writing and sending responses to slide-based activities using digital ink. One student wrote about the experience, “I don’t normally speak up in class... Classroom Presenter allows me to express my ideas but without speaking up in front of everyone.” Another one added, “Some questions are easier to discuss by speaking up, but most of the time I prefer to use the Classroom Presenter.” Yet another student expressed the appreciation among the more vocal students too, noting, “I really liked how everyone had to contribute via the tablets because you get the opinions of the shy people.”

*Reflective writing assignments* have had a long and difficult, but ultimately successful, path toward acceptance and appreciation by students. In earlier offerings, the class-average ratings of the concept of reflective essays started from a low of -0.12, to climb successively through 0.57, 0.44, 0.83, 0.88 to 1.29 (Figure 2). At the same time, the latest format of reflective writings (where the instructors write an extensive response to each student essay and as part of that pose related personalized questions, to which the student responds in turn) underwent a transformation in student opinion from 0.57 to 0.63, 1.13, 1.38, and, finally, 1.43. While it does take instructors about 10-15 minutes per response, it adds tremendous value to student learning, making it well worth the investment of time. Finally, the quality of our individual feedback to student writings received ratings that plot a similar upward trajectory: 0.84, 0.84, 1.05, 1.03, 1.37, 1.60, and 1.50.

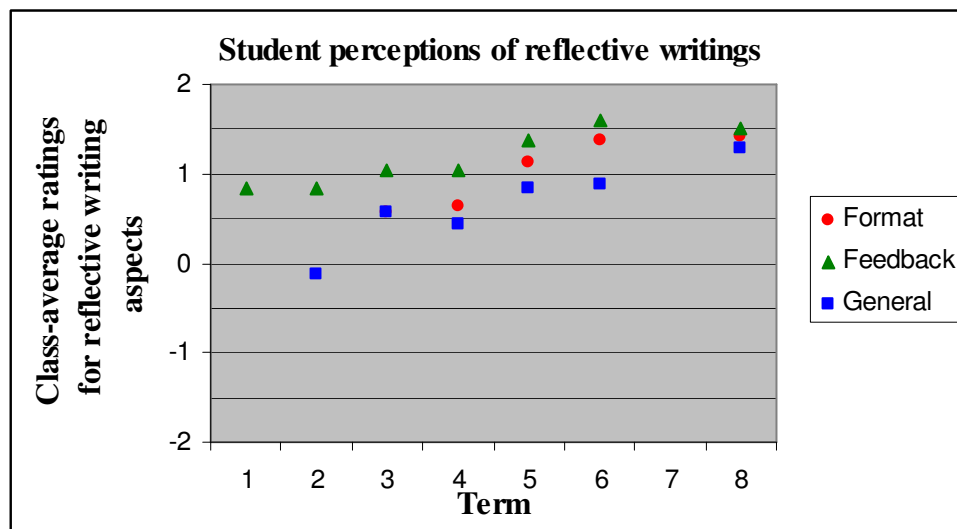
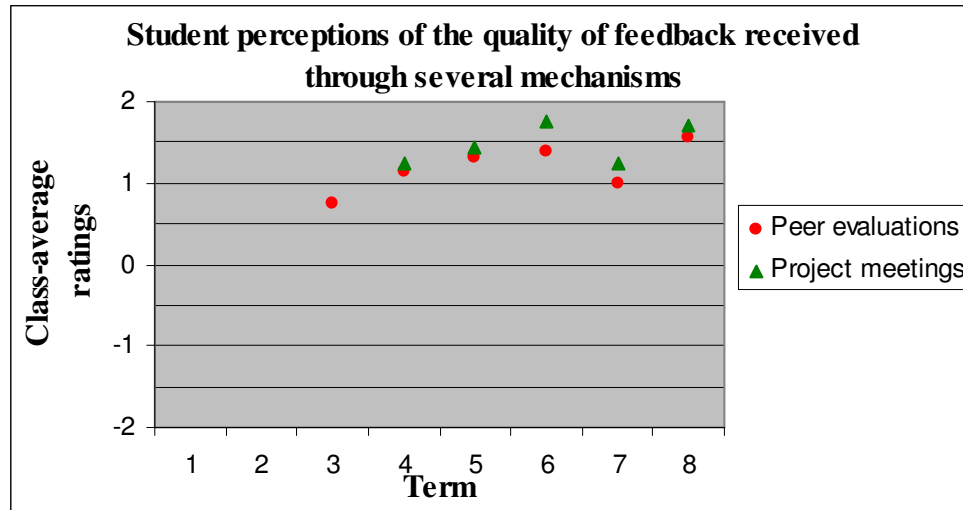


Figure 2. Student perception of aspects of reflective writings – on the general concept, on the format (including responses to instructor feedback), and on the quality of instructor feedback. Missing data points from one of the series accounts for the fact that in some offerings certain aspects were not featured.

Judged on all three aspects, the reflective writing assignments have become one of the students’ favorite components of the course. Indeed, a student wrote that they appreciated essay questions, “because it asks what we personally experience, not what’s [in the book].” Another student, in an earlier course, found the virtual “conversation” with the instructor stimulating and wrote, “The part [of this class] I looked forward to the most besides coding was getting my reflective essay back. [The feedback I got] inspired new thinking.”<sup>7</sup> More recently, a free-form student response on writings in general said, “I was having fun (and thinking a lot) writing individual assignments...”, while another student wrote, “The written assignments were very helpful. Just the right length to make sure I wasn’t missing any important topics.” We believe that the earlier problems with the image of reflective essays were a reflection of several factors, all of which were subsequently addressed: the much higher number of reflective writings per term, the lack of sufficient explicit motivation of the concept, problems with the grading rubric and how students interpreted it, as well as the exclusive presence of “soft” (non-technical) questions on all writing assignments during the first few terms.

The strongly positive perceptions of the usefulness of various forms of feedback applied to other components of the course too: the *post-delivery peer evaluations* and the *post-delivery informal project discussion meetings*. For the former, class-averaged ratings improved from 0.75 to 1.15, 1.31, 1.38, 1.00, and finally 1.57, whereas for the latter the ratings moved from 1.23 to 1.43, 1.75, 1.25, and eventually to 1.71 (Figure 3).



**Figure 3.** Student perceptions of the quality of feedback received through peer evaluations and informal project discussion meetings with instructors after each project milestone.

One metric of success for us is whether *students felt better prepared for industry careers* after taking our course. On that question, almost 88% responded affirmatively. Highlights among the free-form student responses included: “After taking this class, I just don’t see how anyone can be prepared for the industry if they haven’t taken this course.”, “It is likely the most applicable course to the real industry provided in the entire department.”, and “[The project proposal and architecture design] skills were really helpful and something an internship can’t give you.”

A related metric is how much *confidence students gain* in the discipline as they accumulate project experience. The data shows that even within the confines of a single term there is a perceptible difference: the class-average rating of the quality of the students’ own final release products is substantially higher than the rating of their first releases, with a margin of 0.48-0.80.

On the first day of class, we often asked the students to briefly write about “What is Software Engineering to me?”, and promised that their responses will not be used for grading purposes. We collected and kept the responses during the term, returning them to the authors on the last day of class, before students filled out the questionnaire. On the questionnaire we asked them whether their own earlier answer to the above question *differed significantly from their current understanding*. 38% of the respondents in the 4 course offerings when this was done answered affirmatively, showing that a sizable number of the students appreciated the lessons from the course even without the benefit of subsequent industry experience to confirm those lessons.

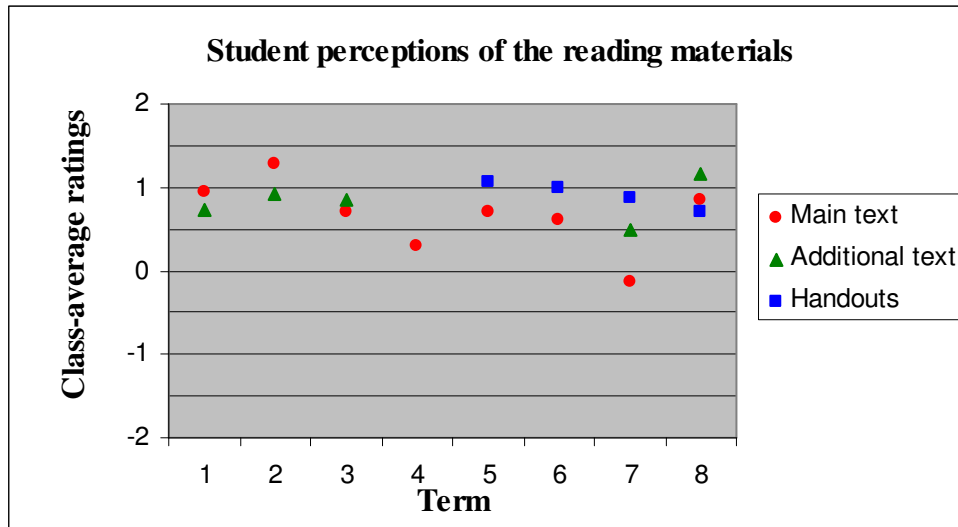
Finally, when we asked students to sum up in *one word* the essence of the course for them, by far the most frequent word choices were “experience” and “teamwork” (even if students had

negative views on certain aspects of the course, as was the case in one of the earlier offerings), followed by “people”, “learning”, “process”, “valuable/useful”, “communication”, “frustration”.

### 3.1.2. What Did Not Work Well

Over time, the list of aspects that “do not work well” has been steadily dwindling. Below is a discussion of the issues that are yet to be resolved.

The *course-related readings* have been a chronic weakness (Figure 4). Standard texts either lack the breadth we aim to cover – including both technical and non-technical aspects – or do not fit well with the experiential nature of our course and its mandate to focus on pragmatic issues above all. In addition, texts like “Rapid Development”<sup>5</sup> have appeared too verbose for students and in some parts assume prior industry experience from readers. That text’s ratings have been lukewarm, albeit generally on the upslope lately (0.31, 0.62, -0.12, and 0.86). Finally, a perceived disconnect between lectures, readings, and project needs – reported by a number of students in the earlier offerings – meant that “there wasn’t enough incentive to complete the readings,” as one student put it. Our latest approach to solving this problem has been to adopt the use of targeted handouts – shorter, practical, and to-the-point (e.g., Joel Spolsky’s “The Joel Test: 12 Steps to Better Code”<sup>8</sup>) – as a supplement to a less complete but more palatable text such as “The Pragmatic Programmer,”<sup>3</sup> offering a recipe-style, textbook-like presentation of material that is practical in nature, so students can more easily relate to it. That book has been generally liked (rated 0.95, 1.28, 0.71, 0.50, and 1.17 in the terms when it was used) but the specific handouts have been met with mixed enthusiasm (rated 1.07, 1.00, 0.88 and 0.71).



**Figure 4.** Student perceptions of the value of the reading materials used in class. Missing data points from one of the series accounts for the fact that in some offerings we did not use an additional text and/or targeted handouts.

Other problematic areas are *quality assurance and testing, build processes, and documentation*. Table 1 shows the class-average ratings for those over the 8 terms.

Table 1. Class-average ratings of problematic areas over all 8 terms. “n/a” in a given cell indicates that we did not ask about the corresponding aspect in that term’s questionnaire.

	Term 1	Term 2	Term 3	Term 4	Term 5	Term 6	Term 7	Term 8
Test plan	n/a	n/a	n/a	n/a	0.46	-0.14	n/a	1.14
Test-driven development	0.59	0.24	0.45	0.68	0.35	0.25	n/a	0.14
Unit testing	1.05	0.92	0.04	0.58	0.35	0.14	0.60	0.86
Acceptance testing	0.27	0.72	n/a	0.79	0.41	0.62	0.83	0.86
Defect tracking	0.27	n/a	1.08	0.95	1.00	0.25	1.17	-0.14
Automated (1-step) build	1.55	0.88	n/a	n/a	n/a	0.86	0.50	0.71
Documentation	0.73	0.51	0.29	1.02	0.51	0.50	n/a	0.93

The table data shows no obvious trend for any of the above areas (with the possible exception of ‘Acceptance Testing’), while most areas have at some point shown troubling signs. As before, lower ratings likely indicate that the corresponding technique/tool was not seamlessly incorporated into the development process for that particular course offering and/or the students did not have initial positive experiences with that aspect<sup>6</sup>. Since most of these techniques/tools are novel for most students, we believe that it is important to devote extra time to their coverage from the very start in order to make adoption possible. It is also important to create a need for their use (e.g., by requiring them as part of a project deliverable), so that they do not merely remain a good idea in theory. As one student wrote, “All those cool testing techniques, we didn’t use them. We didn’t really see a need.”

Although the feedback and lessons from earlier terms were known to subsequent instructors – we were disciplined about examining the data and performing retrospectives before moving on – one factor confounding the interpretation of the data in Table 1 in comparisons between terms is that the instructors changed between terms (with the exception of the author, who carried forward the “messages” from one offering to the next) and their teaching styles often differed. This may have additionally contributed to the lack of visible positive trends in those troubling areas.

*Dealing with ambiguity* is another aspect, central to the profession of software engineering, that students traditionally have difficulty learning. Since most courses in the Computer Science curriculum intentionally offer little or no ambiguity, most students’ natural inclination is to demand precise directions and specifications from instructors – this is what they have been trained to expect from us. Still, there are those (few) students who feel at ease with ill-defined requirements and open-ended problems. Here are quotes that represent both camps: “It wasn’t exactly clear what was expected... sometimes I wasn’t sure whether to add more detail... In the end, I didn’t feel as confident as I should have.” versus “If you spoon feed [students]..., then you lose out on the process... in the field, you won’t have people holding your hand.” Potential solutions to this general problem, of which we are seeing the symptoms in our course, go beyond the scope of any single course and need to be implemented at the program curriculum level.

Finally, *dealing with people* was another uncomfortable situation that students had to endure. They overwhelmingly reported wanting to receive more constructive criticism from peers through the peer evaluations, yet very few ventured to provide honest, direct, and respectful remarks of constructive criticism to their teammates. One student expressed the difficulty thus: “It seemed hard to give criticism in a nice way,” while another commented: “Everyone was entirely too caught up with feelings. You don’t have to be nice. We’re big kids now. We can be told we’re doing something poorly and not cry to our mothers.” To begin addressing the discomfort that many were experiencing, our solution for the long term is to teach students how to provide good feedback to each other. Asking them to participate in peer evaluations several times during the term – both as feedback givers and feedback recipients – is an implementation of that solution idea.

### 3.1.3. What Surprised Us

While all findings reported so far remained within the realm of what we were prepared for, the responses to a few questions took us by surprise.

The vast majority of students (96%) reported *no transfer of ideas* across project teams, even though we intended to encourage them to learn from each other through several elements of the course organization: having in-class presentations/demos of intermediate project deliveries for each team, asking students to provide feedback on each other’s presentations, and (in one course offering) even requiring that all projects have the same high-level theme (we chose “remote collaboration” as the theme). One possible explanation of the reported lack of transfer may be the students’ desire to stay clean on any potential suspicions of cheating. Another is that they may have misinterpreted our question “Did your team borrow ideas from what the other teams have done?” as an under-cover attempt to find out if their team had cheated. Given the very good rapport that we had developed with the students in those classes, we find these explanations hard to believe. A more plausible scenario from our perspective is that in their desire to firmly establish their team’s identity and distinguish themselves from others in the class, students may have shunned the opportunities to take advantage of ideas that they could not call their own. Teams did, however, frequently entertain the thought of incorporating open-source modules into their projects, and sometimes acted on those prospects.

Another surprise to us was that the majority of the students independently reported that “*I don’t like watching myself on video* [speaking in front of an audience],” despite the obvious learning benefits of doing so. One student elaborated, “I wasn’t forced to. Watching myself speak in public on camera takes extreme courage... That might be a good idea though because I think it would probably be a very revealing and educational, if traumatic, experience.” One way to take advantage of the learning opportunities in students observing themselves from a new angle may be by posing a related question on a reflective writing assignment – a question that forces students to confront their fears and write about what they learned from the experience. Needless to say, this must be done very gently and supportively, to avoid alienating students.

The last surprise came in response to: “What types of questions and/or topics did you feel least prepared to answer on the final exam?” (For this one term, the final exam was done in class, rather than as a take-home exam.) A sizable percentage of the respondents said they were *least*

*prepared for questions that directly related to the readings and materials discussed in class. We attribute such responses to the likely scenario that many students were so focused on their project work that they largely dismissed any seemingly unrelated readings and classroom discussions.*

### **3.2. What We Learned about the Process of Surveying Students**

As with every major undertaking, one has a chance to learn not only about the subject of inquiry – the software engineering course, in our case – but also about the very process of conducting this inquiry, including how it could be improved. Here we report our findings about this process.

#### **3.2.1. The Evolution**

The student questionnaire has evolved significantly since its inception 4 years ago. It grew from a total of about 60 questions (both multiple-choice and free-form) in its initial version to around 150 questions today. As we strived to improve the course, we had new questions that needed answers; those would have been very useful questions at the start too, but we had not discovered them yet. Occasionally, some questions would be retired as we realized that the responses we were receiving did not offer us sufficient distinguishing power or any deep insights. Removing old questions was also the result of pressure to keep the questionnaire to a manageable size – it had to remain such that students could reasonably complete it in about 30 minutes.

Another change in the process of conducting questionnaires is in the medium. The first two terms it was done on paper, but the aggregation of results – a routine step in itself – turned into a major investment of time because of the sheer number of questions and the number of students in our courses: there were over 1000 line items to look at in the results of each questionnaire. Today, questionnaires are done online and the aggregation is performed by a script we have developed, producing a report that a human can more easily analyze.

In examining the collected data, we discovered that students had specific and interesting comments mostly about recent events and class sessions. Asking detailed questions about aspects from the more distant corners of their memory (e.g., referring to specific class sessions from over a month ago) – almost regardless of the questions asked – tended to result in similar ranges of responses, and similar, unenthusiastic class-average ratings. We called this a “default range” of answers. Those default range-producing questions were useful too, however, especially in providing a baseline of comparison with the responses to other questions that mattered.

Finally, in the process of analyzing the questions, we distilled three groups of them, each group serving a distinct goal. One group consisted of questions that would be especially useful for deciding how to evolve a course; another group would aid in understanding the mood of the respondents and thus calibrating the results of a given questionnaire; and yet another would be effective at getting the respondents to reflect on specific aspects of the course and to appreciate what they have learned in the course. Not surprisingly, most of these are free-form, short-answer questions.

They are also valuable questions to choose from and pose in a brief mid-term questionnaire, aimed at one or more of the three goals: discovering what works and what does not in order to help instructors be more effective for their students starting in the current course; discovering student attitudes toward the course or any specific aspects of it; or directing students' thoughts toward potentially fruitful areas of reflection. One of our favorite multi-purpose questions is: "I could extract more value out of this course by ... (complete the sentence)."

Next, we list the questions from each of the three groups. Within each group questions appear in no special order.

### 3.2.2. Useful Questions for Evolving the Course

- "What aspects of this course would you have liked the instructors to spend more/less time on?"
- "I could have extracted more value out of this course by ... (complete the sentence)."
- "Which lectures were most memorable and useful to you?"
- "For the lectures that did not 'click' well for you, what might have made them more useful?"
- "Would you have appreciated more constructive criticism from course staff [on the peer reviews]?"
- "Was there ever an important issue to you related to this course that you felt uncomfortable sharing with the course staff? If so, what might we do in the future to make it easier for students to be comfortable sharing such concerns?"
- "What was the average number of hours per week you put into this course? Is this more than you had planned initially, or less, or about the same?"
- "Of all the activities you did as part of this class, which ones (if any) do you feel were not as valuable in advancing your education? Among them, which would you suggest keeping in future courses but in a different form? What change would you suggest?"
- "What types of questions and/or topics did you feel least prepared to answer on the final exam?"
- "Do you feel you could have benefited from hands-on tutorials on certain topics? If so, which topics?"
- "Are there useful types of feedback that were not utilized at all in this course? If so, what are they?"
- (Also, any seemingly innocuous questions that have revealed surprising responses are good candidates for elaborating on in future questionnaires – to find out if attitudes have since changed and if not, to learn what exactly is going opposite to our intuition.)

Out of the student responses to the first question on this list, we as instructors learned to be more concise in our presentations, to consciously and explicitly connect readings to current projects, and to not hesitate to relate stories of our own industrial experience doing software engineering.

### 3.2.3. Useful Questions for "Reading" the Students' Moods

- "Did we manage to address the concerns you brought up in the anonymous midterm questionnaire [since then]?"

- “The distribution of work within your team” (Note to readers: this is a multiple-choice question.)
- “Would you recommend this course to others?”
- “Was the time you spent justified by the value you got from [the most time-consuming activities in the course]?”
- “If you had to sum up in *one word* the essence of this course for you, what would that word be?”

#### 3.2.4. Useful Questions for Getting Students to Reflect

- “Does your answer [from the beginning of the course] to ‘What is Software Engineering to me?’ differ significantly from your current understanding?”
- “I could have extracted more value out of this course by ... (complete the sentence).”
- “Did the [peer evaluation] results change the dynamics within your team?”
- “Overall, how do you feel about what the course offered in relation to your objectives?”
- “Are you satisfied with what you learned in this course?”
- “Do you feel that what you learned in this course will change what you do in future projects?”
- “What aspects from this course do you hope to be able to use in other courses / projects, whether software-related or not?”
- “What two activities (both in and out of class) consumed most of your time in this course?”
- “(To quote Brooks from ‘The Mythical Man-Month’<sup>2</sup>), write one thing from the course that you will surely remember for the rest of your life.”
- “What are some good motivational strategies you’ve found to work well in a situation when a teammate was not motivated? What were some techniques that you tried but that did not work well, even though you at first expected them to be effective?”
- “Have you helped others in the class learn something new? Have others in the class helped you learn something new?”

As if to confirm the reflective value of these questions to students themselves (in addition to the pure feedback value to instructors of the questionnaire as a whole), a student recently wrote, “[The questionnaire] was useful for me to look at what I liked/disliked in this course.”

## 4. Summary and Conclusions

The main contributions of this work are the analysis of a large body of data from extensive course-specific student questionnaires conducted in a software engineering course, as well as the distilling of groups of questions that are especially useful for evolving a project-based course, for “reading” students’ moods, and for getting students to reflect on their own experiences.

In the paper we explored course-related aspects that did work well, as well as aspects that did not work well, and even some that took us by surprise. Our conclusions are grounded in our rich feedback data set.

One of the highlights for us as instructors is the finding that our students overwhelmingly report feeling better prepared for industry careers after taking the course.

The results of this paper may be of interest to instructors of software engineering, as well as, more generally, to engineering educators who teach project-based courses or simply want to measure student attitudes toward aspects of their courses.

## Acknowledgements

The author would like to thank his colleagues Gail Alverson, Richard Anderson, James Bullock, Douglas Johnson, and David Socha for the many productive discussions around the course over the years. The help of Stani Vlasseva in proof-reading and making constructive suggestions to earlier drafts of this paper was invaluable and much appreciated. Last but not least, without the helpful feedback shared by our many dedicated students this work would not have been possible.

## References

- [1] Richard Anderson, Ruth Anderson, Oliver Chung, K.M. Davis, Peter Davis, Craig Prince, Valentin Razmov, and Beth Simon. "Classroom Presenter - A Classroom Interaction System for Active and Collaborative Learning." In WIPTE, Apr. 2006.
- [2] Fred Brooks, *The Mythical Man-Month (20<sup>th</sup> Anniversary Edition)*. Addison-Wesley, 1995.
- [3] Andrew Hunt, David Thomas. *The Pragmatic Programmer: From Journeyman to Master*. Addison-Wesley, 2000.
- [4] Norman L. Kerth. *Project Retrospectives: A Handbook for Team Reviews*. Dorset House Publishing, 2001.
- [5] Steve McConnell. *Rapid Development*. Microsoft Press, 1996.
- [6] Philip Ross. "The Expert Mind." *Scientific American*, July 24, 2006.
- [7] David Socha, Valentin Razmov, Elizabeth Davis. "Teaching Reflective Skills in an Engineering Course." In ASEE, Jun. 2003.
- [8] Joel Spolsky. "The Joel Test: 12 Steps to Better Code." Available at <http://www.joelonsoftware.com/articles/fog0000000043.html> .

## Appendix: Full list of questions from the most recent end-of-term student questionnaire

Legend: <sup>MC</sup> = multiple choice question; <sup>YN</sup> = yes-no question; <sup>FF</sup> = free-form question

### Development System

- <sup>MC</sup> Teams of size 4 students (instead of smaller teams, like many other CS courses have)
- <sup>MC</sup> Different teams working on different project ideas (as opposed to working on the same project)
- <sup>MC</sup> The project idea that your team developed
- <sup>MC</sup> The shared project space of your team (e.g., wiki pages, repositories, etc.)
- <sup>MC</sup> The official course website: announcements, course schedule (with slides and videos) resources, milestone submissions
- <sup>FF</sup> Which portion(s) of the course website were most useful to you?
- <sup>MC</sup> The class mailing list
- <sup>MC</sup> Your team's mailing list (or other electronic communication medium you used)
- <sup>MC</sup> Your team's project schedule
- <sup>MC</sup> Your team's project specification
- <sup>MC</sup> Your team's architecture and design documents
- <sup>MC</sup> Your team's test plan
- <sup>MC</sup> Your team's zero-feature release product
- <sup>MC</sup> Your team's beta release product
- <sup>MC</sup> Your team's final release product
- <sup>MC</sup> The other team's milestone releases
- <sup>MC</sup> The other team's final product
- <sup>YN</sup> Did your team borrow ideas from what the other team had done?
- <sup>MC</sup> Your team's *ongoing* technical documentation
- <sup>MC</sup> Your team's *final* technical documentation
- <sup>MC</sup> Your team's *final* user documentation
- <sup>MC</sup> Your team's project meetings (weekly and other)
- <sup>MC</sup> Forming teams through gathering and then matching everyone's preferences for a proposed set of project ideas
- <sup>FF</sup> If you have alternative suggestions for how to more effectively form teams, what are they?
- <sup>MC</sup> The development environment you used (e.g., Eclipse, etc.)
- <sup>MC</sup> The use of a configuration management system (e.g., CVS, Subversion)
- <sup>MC</sup> Unit testing (e.g., via JUnit, etc.)
- <sup>MC</sup> Acceptance/system testing
- <sup>MC</sup> Defect tracking (e.g., via Bugzilla, etc.)
- <sup>FF</sup> If there were resources you needed / would have benefited having that were not available to you (software or hardware, technical or non-technical materials via the course web's 'Resource' page, etc.), what were these resources?
- <sup>MC</sup> The incremental release approach (zero-feature release → beta release → final release) to product development
- <sup>MC</sup> One-step build
- <sup>MC</sup> Test-driven development
- <sup>MC</sup> Using Classroom Presenter to allow the instructor to dynamically annotate slides during lectures

- MC Using Classroom Presenter to actively involve you in class through student submission activities
- FF Do you prefer to participate in class via Classroom Presenter's student submission activities or do you find more value in directly participating in discussion-style lectures (where no student devices are used), e.g., by raising your hand or speaking up?
- YN Do you feel you could have benefited from hands-on tutorials on certain topics?
- FF If "yes", which topics?
- FF What are your thoughts about using your project ideas vs. if *we* had given each team a project to work on?
- FF Other comments on the development system:

### Readings

- MC Reading "Rapid Development"
- MC Reading "The Pragmatic Programmer"
- MC Reading "Death March"
- MC Reading additional suggested books, articles, and handouts on special topics
- MC The quality of the handouts we distributed
- FF Which of additional suggested materials (books, articles, handouts) did you find most useful for your own learning?
- FF Other comments on readings:

### Writings

- MC Individual assignments (in general)
- MC The format of individual assignments: writing answers and later responding to our comments and questions on your assignments vs. if there were more assignments but without follow-up responses to our questions
- MC The format of our responses to your assignments: in electronic form instead of on paper
- MC The overall quality of the questions on the individual assignments: Were they thought-provoking? Did you learn something useful for yourself as a result of reflecting on and answering those questions?
- YN Would you have preferred to be able to see the writings (likely anonymized) of other students in the class?
- YN Would you have been comfortable writing what you wrote in your own answers if other students could read it (and assuming that your writing was anonymized)?
- MC The quality of the questions on the midterm exam
- MC The quality of the sample solutions to the midterm exam questions
- MC The quality of the questions on the final exam
- MC The quality of former midterm and final exams (e.g., from summer 2005 and/or spring 2006) as a tool for exam preparation
- FF Roughly, how much time did completing the take-home final exam actually take you?
- YN Does your initial answer to "What is software engineering to me?" differ significantly from your current understanding?
- FF Other comments on writings:

### **Class Sessions (lectures, sections, discussions, guest lectures, presentations)**

- MC The style of teaching / learning through experience (where students are responsible for all phases of the project throughout the lifecycle)
- MC Having guests come to our classes and contribute their ideas and opinions
- MC Having guests come to your presentations/demos to give you feedback as an outside perspective
- MC Having guests give lectures (in general)
- MC The opportunity to network with outside professionals in the field
- MC A's guest lecture on "ABC"
- MC B's guest lecture on "KLM"
- MC C's guest lecture on "XYZ"
- YN Would you have preferred to see more guest lectures?
- MC Our coverage of technical software engineering topics
- MC Our coverage of "softer" (non-technical) software engineering topics
- MC Proposing a list of advanced topics (around week 7) and soliciting your opinion on which ones you'd most like to see covered
- MC The instructor's lecture content
- MC The instructor's presentation style (through discussion and student involvement)
- MC The TA's lecture content
- MC The TA's presentation style
- FF If you would have preferred more interactive activities in lectures and/or sections, what types of activities would have helped you the most?
- FF Which of the instructor's lectures were most memorable and useful to you?
- FF Which of the TA's lectures were most memorable and useful to you?
- FF For the lectures that did not "click" well for you, what might have made them more useful?
- MC Your project proposal presentation
- MC Your team's architectural design presentation
- MC Your team's zero-feature release presentation/demo
- MC Your team's beta release presentation/demo
- MC Your team's final release presentation/demo
- MC The other team's presentations/demos
- FF Other comments on class sessions:

### **Feedback**

- MC The instructor's feedback on your assignments (#2)
- MC The TA's feedback on your assignments (#1 and partly #2)
- MC The instructor's comments and instructions sent to the mailing list
- MC The TA's comments and instructions sent to the mailing list
- MC The instructor's feedback to your team (through email, discussions, etc.)
- MC The TA's feedback to your team (through email, discussions, etc.)
- MC The instructor's individual feedback to you
- MC The TA's individual feedback to you
- MC The instructor's in-class feedback to student submissions (gathered using Classroom Presenter)
- MC The informal team discussions with course staff (following each milestone) as a way of giving your team feedback on the project

- MC The feedback by course staff on your project proposal
- MC The feedback by course staff on your team's architectural design
- MC The feedback by course staff on your team's zero-feature release
- MC The feedback by course staff on your team's beta release
- MC The feedback by course staff on your team's final release
- MC The feedback by guests during in-class presentations/demos
- MC The feedback by students from the other team (at the project proposal, beta release, and final release stages)
- MC The outside feedback you sought (and received) on the usability of your product
- MC The direct feedback from teammates on your own project work and participation
- MC The in-class retrospectives after each project milestone
- MC The format of retrospectives: "sustain" and "improve" categories
- MC The peer evaluations as a form of feedback
- MC The value *to you* of peer evaluations given by your teammates
- MC The value *to you* of peer evaluations given by students from the other team (if available)
- MC The value *to you* of peer evaluations given by course staff
- MC Incorporating peer review results (except the first peer review) into grading as a way to ensure accountability
- YN Would you have appreciated more (constructive) criticism from teammates on the peer evaluations?
- YN Did you provide constructive criticism to teammates in the peer evaluations you wrote?
- YN Would you have appreciated more (constructive) criticism from course staff on the peer evaluations?
- YN Did the peer evaluation results change the dynamics within your team?
- FF What type of feedback would you prefer to see more of? (Constructive criticism? Positive reinforcement? Other?)
- YN Are there useful types of feedback (to you and your classmates) that were not utilized at all in this course?
- FF If "yes", what are they?
- YN Was there ever an important issue to you related to this course that you felt uncomfortable sharing with the course staff?
- FF If "yes", what might we do in the future to make it easier for students to be comfortable sharing such concerns?
- FF In the anonymous midterm questionnaire we asked what we could be doing better for the then-remaining few weeks in the quarter. Did we manage to address the concerns you brought up?
- FF Other comments on feedback:

### **Miscellaneous**

- MC Your project team (as a whole)
- MC The distribution of work within your team
- MC Having 1-hour class sessions four times a week (as opposed to, say, 2-hour sessions twice a week)
- MC The classroom setup (allowing for frequent use of technology, two public displays, lectern in the middle, etc.)
- FF What was the average number of hours per week you put into this course?

- FF Is this more than you had planned initially or less, or about the same?
- YN Did you watch (or listen to) any of the videos of lectures?
- FF If not, why not?
- FF If yes, did you do this regularly or only when you were absent from class or needed to revisit specific content, e.g., while preparing for an exam?
- YN Did you watch (or listen to) yourself on the videos from the various presentations/demos?
- FF If not, why not?
- FF What aspects of this course would you have liked the instructor to have spent *more* time on?
- FF What aspects of this course would you have liked the instructor to have spent *less* time on?
- FF What aspects of this course would you have liked the TA to have spent *more* time on?
- FF What aspects of this course would you have liked the TA to have spent *less* time on?
- FF What topics (or classes of topics) would you like to have seen *more* of in this course?
- FF What topics (or classes of topics) would you like to have seen *less* of in this course?
- YN Have you helped others in the class learn something new?
- YN Have others in the class helped you learn something new?
- MC Overall, how do you feel about what this course offered in relation to your learning objectives?
- YN I realized that this course was not the right place for the things I wanted to learn.
- YN I changed my initial objective during the course, and now I feel satisfied with the achieved results.
- YN I realize that the course offered more opportunities to learn than I took advantage of.
- YN I took all I could in exactly the way I had anticipated.
- FF I could have extracted more value out of this course by ...
- FF Other? (please specify, if any):
- YN Do you feel better prepared for working in industry after taking this course?
- YN Would you recommend this course to other students?
- FF Other comments in the miscellaneous category:

### Free-form short-answer questions

- FF Are you satisfied with what you learned in this course? Please explain in 1-2 sentences.
- FF Do you feel that what you learned in this course will change what you do in future projects? Please explain in 1-2 sentences.
- FF What aspects from this course do you hope to be able to use in other courses / projects (whether software-related or not)?
- FF Of all the activities that you did as part of this class, which ones (if any) do you feel were not as valuable in advancing your education? Among them, which would you suggest keeping in future courses but in a different form? What change would you suggest?
- FF What two activities (both in and out of class) consumed most of your time in this course?
- YN Was the time you spent justified by the value you got from them?
- FF (To quote Brooks from 'The Mythical Man-Month') Write one thing from the course that you will surely remember for the rest of your life.
- FF If you had to sum up in *one word* the essence of this course for you, what would that word be?

### Final question

- FF What do you think about this questionnaire?