Order Out Of Chaos: A Table to Help the Design of Project-Based Courses

David Socha[§†], Valentin Razmov[§]
[§] Department of Computer Science & Engineering
[†] Center for Urban Simulation and Policy Analysis
University of Washington, Seattle
{ socha, valentin } @ cs.washington.edu

Abstract

The project-based software engineering course that we teach uses several different teaching methods to instruct students in a large number of reflective and analytical techniques. These techniques help the students learn how to work in teams and on projects. As we, instructors, were preparing to teach the course for a third time, we had to sort out a confusion in the course design, brought about by the presence of many techniques taught in many ways. We devised a method to organize these techniques by scope (for individuals, teams, projects, systems) on one axis, and by how they were taught (as mini-lectures, homework assignments, project experience, coaching sessions, experiential sessions, etc.) on another axis. This enabled us to see "holes" in our course design that were not obvious before. As a result, we adjusted our priorities accordingly and focused our efforts. This paper shows how we evolved the course structure, discusses how this process helped and surprised us, and speculates about how the structure may be applied to other courses that wish to create a multi-faceted learning environment.

1. Introduction and Context

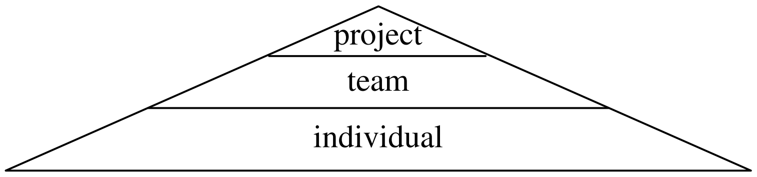
Our goal when teaching software engineering is to educate the students to appreciate the importance of the human aspects of software development. In particular, our industry experience indicates that software engineering is characterized by people working together under pressure to deliver value to their customers.

Our tactic for teaching this is to create an environment that simulates an industrial experience, but where the success metric is how much students learn, not whether the project they deliver is "successful." In particular, we have students organize into large project teams (of 15 or more). These large teams usually lead to the students adopting a hierarchical organization, with leadership roles and component sub-teams within the project team. As a result, students encounter many of the inter-personal issues that exist in all workplaces, and are forced to deal with the gamut of team and project coordination issues that determine the success or failure of virtually all projects.

An important aspect for the success of this type of course is to make sure that the students realize how much they have learned despite the frustrations of dealing with the complexities and emotional issues of large teams. Thus, our course emphasizes teaching a variety of reflective techniques[6,4] so that students (and instructors) can identify what they have learned, including what they now know that they still need to learn. We know of other courses that used large project teams, but without such a reflective feedback mechanism – which resulted in students who were bitter and disappointed at the end of the course.

In our course we also emphasize experiential learning[8] and guided discovery[3]. It is well known that students learn much better by doing than by passively listening. Experiential learning is a two-step technique that starts with putting students into situations where they discover (through reflection) that their current mental models and skills prevent them from getting what they desire. Once this is achieved, the second step involves guiding the students by introducing new models that may allow them, with practice, to attain their goals. Succinctly put, students do more and instructors talk less in comparison to the conventional lecture-style of teaching. Experiential simulations offer a form of experiential learning where students participate (the "doing" part) in a "game," followed by debriefing sessions (the reflective part) to extract the lessons that emerge from what they have done while in the game. A term-long project provides additional experiential material, while weekly reflective writings help the students exercise their reflective skills and discover what they have learned.

Last spring, as we, instructors, were discussing the many (19 at the time) reflective techniques that we had used in our course[6], we realized that each technique was working on one of three different levels - individual, team, or project. We also realized that our course had more individual techniques than team techniques, and more team techniques than project techniques. This can be illustrated by the following pyramid structure:



We believe this "imbalance" is desirable for several reasons. Individual practices form the basis for individual learning. They tend to have more immediate value to students. A student can practice them alone and without the need for buy-in or cooperation from anyone else. Individual techniques can be used in any domain, and are key for enabling students to be life-long learners[2,5]. Many of the individual skills make it easier to learn the group skills, since through their practice the individual becomes more aware of how she effects the team. Team techniques can be practiced within a sub-team, instead of within the larger, and thus potentially less safe, project team. These group skills in turn make it easier to learn project skills, since large projects are fulfilled by teams working together.

This paper describes how we devised a table that allowed us to see how well the practices "covered" each of the levels and to assess whether each collection of practices within a single level was appropriately balanced. In our case, we felt that we had enough coverage and balance at the time.

2. Restructuring the Course

Four months later, last autumn, we met to organize our third offering of this course. Instead of having 2-hour or 3-hour long sessions, as we had had before, we would be constrained to 50-minute class periods. Also, instead of 20-some students, we were told to expect up to 60 students. These administrative changes required that we significantly restructure our course. For instance, experiential simulations require at least 2 continuous hours to be effective.

We needed to decide which aspects of the course to keep, which to modify, which to discard, and which to add. In the process, we listed all reflective practices and domain-specific content modules we wanted to convey, roughly 40 in all. The result was hard to make sense of … until one of us suggested that we map out the desired items similarly to how we had done with the pyramid before. This time we used a different representation (but the same semantics), listing each item into a table with two columns: Level and Item.

While filling in the table, we realized that in addition to individual, team, and project, there was a fourth "level," or domain, of techniques: systems. Our systems level contained techniques and material applicable to any system[9], including individuals, teams, or projects. These techniques include the scientific method, test driven development[1], and diagrams of effects[9]. Adding the separate systems category validated our intuitive, but not explicitly recognized, emphasis on system techniques in the course.

The table helped us to assess the coverage of the techniques in our course. At that point we needed to determine which of the things previously covered by experiential simulations could be conveyed effectively in other ways within the constraints of 50-minute class sessions and with potentially twice as many students to guide. This required additional columns in the table to reflect the different manners in which we would teach each item. We used the following extra columns (shown here with their respective meanings):

- S = requires a full 50-minute class session
- H = can be given as homework
- L / l = works well as a lecture / mini-lecture
- P = can be derived from working on the project
- E = an experiential session
- M = miscellaneous; anything not covered by one of the other categories

Some items we felt we could convey to the students through several different approaches, so they were assigned to multiple categories.

Such a single-page view of the entire course structure (see Appendix A for an example) led to several important realizations:

- There was lack of homework at the system and project levels. Since homework is a traditional way to force students to become engaged in and practice newly taught material, the lack of it for these levels led us to wonder whether the students would actually learn about the things we would teach at those levels.

  We decided that we did not need homework assignments to reinforce the lessons at the project level. The project work itself served that role well and students were already graded upon how well they managed their project. Our experience from previous course offerings showed us that they would become fully engaged in the project and would drive themselves hard to deliver value. Moreover, they tended to also become engaged in the experiential simulations that addressed project issues.

The lack of homework in the systems area, however, was a problem. Without a forcing function for students to practice the system tools, most of them appeared to have neglected these tools in the past. The "hole" stood out on the table, but we were not sure how to address it effectively given the limited time in the course and our other priorities.

- There were few mechanisms for getting students into the practice of managing their own commitments. If individuals mismanage their commitments, the team may have a problem managing its commitments, which in turn may result in the project having trouble managing its commitments. We still do not have a clear idea how to effectively teach this important aspect of being a professional, while at the same time giving students the freedom and responsibility to navigate the project (a central theme of our course).

- We had too much to cover for the time given. To help determine what to keep, we added a Priority column and started building a tentative course schedule by adding high-priority items first. This resulted in several of the items being dropped because there was no longer room for them on our calendar. However, the decision to drop them was not arbitrary. Many of the practices these items aimed to teach were either not as successful in the previous two offerings of the course or were already covered by one or more other items from the same level, albeit perhaps in pieces.

- To schedule items in particular slots in the course, we needed a calendar view showing what items are to be covered on each day. Ideally, the calendar and the table would be two views of the same data, but we did not have a system to automatically support that. Instead, we added a Scheduled column to the table to indicate which items had been scheduled on the calendar. While flipping frequently between the calendar and table views was inconvenient and there was redundancy between the data in the two, that approach was good enough for our purposes.

  The calendar view facilitated scheduling the fundamental recurrences that drive the project, such as when to: introduce the project, form teams, have students make project deliveries, have students do the final customer presentation, and have project retrospectives. Creating a good structure makes the rest of the course much easier to handle. It also helps set student expectations on the first day of class.

  Once these recurrences were set in the calendar, we filled the remaining class sessions by adding items from the table, starting with the highest priority items. When the calendar was full, we knew which items to include in the course. At this point we checked if anything important was left out and adjusted as necessary.

- As we put material in the calendar, we often agonized over whether to introduce the material earlier or later. Early is good because the students can have more practice using that knowledge during the course. Later is good, since students often ignore our early advice until they have experienced the problems of not using that knowledge[7]. Indeed, our experience shows that some of the lasting lessons students have learned have come as a result of their insistence on doing things their way; when this approach failed them, they realized that their assumptions had been wrong and were happy to adopt alternative models that we suggested.

Appendix A shows a version of this table from the design of our most recent course. While specific items in the table may differ across disciplines, we expect the general approach (of determining item priorities and coverage at all levels) to remain similar.

3. Conclusion

This paper presents a simple tool that helped us to organize our course structure, and see non-obvious patterns, such as places where the structure was insufficient. We believe a similar approach could be useful for other courses, especially courses with a rich structure in which students work in teams on a project.

Acknowledgments

This paper is the result of the collaborative design of our course with James Bullock, Elizabeth Davis, and Stani Vlasseva.

References

[1] Kent Beck. *Test Driven Development.* Addison-Wesley, 2002.
[2] John Bransford (ed.), et al. *How People Learn: Brain, Mind, Experience, and School: Expanded Edition*, 2000, available at http://www.nap.edu/books/0309070368/html/
[3] Ann L. Brown and Joseph C. Campione. *Guided Discovery in a Community of Learners.* In *Classroom Lessons: Integrating Cognitive Theory and Classroom Practices*, K. McGilly (ed.), pp.229-270, MIT Press, Cambridge, MA, 1994.
[4] Valentin Razmov and Stani Vlasseva. *Feedback Techniques for Project-based Courses.* To appear in American Society for Engineering Education (ASEE) Annual Conference and Exposition, 2004.
[5] Donald A. Schön. *The Reflective Practitioner: How Professionals Think in Action.* Basic Books, NY, 1983.
[6] David Socha, Valentin Razmov, and Elizabeth Davis. *Teaching Reflective Skills in an Engineering Course.* In Proc. American Society for Engineering Education (ASEE) Annual Conference and Exposition, 2003.
[7] David Socha, Valentin Razmov, and Elizabeth Davis. *When Conflict Helps Learning.* In Proc. American Society for Engineering Education (ASEE) Annual Conference and Exposition, 2003.
[8] Daniela Weinberg and Gerald M. Weinberg. *Learning by Design: Constructing Experiential Learning Programs.* Readings for Problem Solving Leadership, Weinberg and Weinberg, 1985.
[9] Gerald M. Weinberg. *Quality Software Management, Vol. 1: Systems Thinking.* Dorset House Publishing, 1992.

Biographical Information

DAVID SOCHA, Ph.D.
David Socha studies the human side of software development. He is a Project Manager in the Center for Urban Simulation and Policy Analysis, and a Lecturer in the Computer Science & Engineering department, both at the University of Washington, Seattle, where he received his Ph.D. in 1991. David spent 11 years in industry, 6 of those managing teams of software developers, before returning to academia.

VALENTIN RAZMOV
Valentin Razmov spends time in the classroom as often as he can. He is interested in methods to assess and improve the effectiveness of teaching and learning. Valentin is a Ph.D. candidate in Computer Science & Engineering at the University of Washington (Seattle), where he received his Masters degree in Computer Science in 2001. Prior to this, in 1998, he obtained a Bachelors degree with honors in Computer Science from Sofia University (Bulgaria).

Appendix A

A version of the table we used while designing our course for Winter 2004. Specifics of what each activity constitutes are unimportant here. Significant is the process of determining the priorities and the coverage of different levels (individual, team, project, and systems). While specific activities would differ across disciplines, we expect the process to remain similar.

- S – session
- H – homework
- L/l – lecture/mini-lecture
- P – project
- E – experiential session
- M – miscellaneous
- Priority – should we include this practice in the course? (0: not this time; 1: if time permits; 2: likely; 3: absolutely)

Level	S	H	L/l	P	E	M	Priority	Item	Scheduled
Project					E		3	Good Enough	
					E		3	Testing	Jan 29 / Feb 2 ?
				P			3	Professional Level Tools	
					E		3	Software Process (Waterfall / Spiral / Etc.)	after 2nd delivery ?
			L				3	Project Planning – Commitments Session	scheduled
				P			3	Status (Part Of Project Planning)	scheduled
					E		3	Retrospective	end of course
				P			3	0-Feature Release / Thin Vertical Slice	Jan 9
			l				3	Value (Software Engineering)	
					E		2	Mythical Man Month	
					E		3	Requirements Gathering Session	Jan 15
Team					e		2	Appreciations	
			L		E		3	Speech Acts Session	
			l		E		3	Games And Roles	
			L				1	Ethics	
		H					0	Team Report	
		H					2	Team Conversations (Motivate/Force Them Report)	scheduled
					E		2	Conversation Meter	
		H					0	Team Ranking	
Individual					E		2	What Do You Do When You Don't Know What To Do?	
						M	3	Journaling	scheduled
	S						3	Personality Types	
		H	l				3	Mastery	scheduled
					E		3	What State Am I In?	
		H					3	Portfolio	
		H	l				3	Peer Reviews	
			L				3	Experiential Learning Model	part of the Welcome Session ?
			L				3	Satir's Change Model	
		H					3	Reflective Essays	scheduled
Systems			L				?	Usage-Centered Design	
			l				2	Diagram Of Effects	
			L	p			2	Test Driven Development	Jan 16
			l				3	Scientific Method, Assumptions Session	
			l				3	Configuration Management	
			l	P			3	Software Design	
					E		2	CRC Cards	
					E		0	Card-Storming	