Abstract: Simulating Industry in the Classroom

Joel Weinstein
Northeastern University

Overview:

One of the underlying themes that distinguishes engineering technology from other technical disciplines is the real-world industrial nature and influence over the entire educational experience. While co-op and vacation work activities certainly help to reinforce the industrial "flavor," they are not part of the daily academic regimen.

To provide the industrial experience in classroom activities, we have developed a software project-based course that simulates industry. Because of Rapid Application Development tools that have emerged for the software industry, students can develop and deliver an industrial-strength prototype in a short period of time. There are other project-based courses available, but their emphasis is traditionally placed on delivering a hardware solution—an approach that forces students to focus on time-consuming hardware development activities.

Software projects are different. To be successful, they require a team effort because the overall task is too large for a single student to complete. But modular software development techniques allow student teams to create and manage a series of cooperating tasks that must be managed to be successful. As a result of this requirement, it becomes easy to create a student team hierarchy that simulates a management team and team activities in a traditional industrial setting.

To add further realism to the course, the student teams interface with a disinterested company representative (the instructor) who has been appointed by his fictitious company to complete a task. The representative (deliberately) lacks any detailed knowledge of the solution and is slow to produce necessary information to complete the project. This forces the student teams to make critical decisions on their own because the end-of-course deadline is fixed.

The student teams are assembled randomly insuring that existing friendships do not play a role in the team organization. This more closely simulates an industrial setting where participants are forced to work with existing personnel and resources. The student teams are also prohibited from "firing" a team member because industrially it would cost too much time and money to find a replacement.

To add another dimension of reality, a business school finance major is assigned to each team to assess and enforce loaded cost figures. This forces the team members to think in real financial terms and to more carefully budget their efforts.

The final projects are presented to "the company" (a selected group of faculty) where a formal presentation and a prototype are delivered.

Introduction:

One of the key differentiators between engineering and engineering technology is the "industrial" flavor of the latter. Northeastern University is recognized as one of the pioneers of cooperative education where industrial experience is integrated into the curriculum. But good preparation for the coop experience is necessary if both student and company are to benefit. Over the years, coop jobs have migrated from intrinsically simple tasks to relatively important functions where students actually join a project team and make significant contributions. Preparing them for these kinds of jobs requires more than the teaching of technical skills. Students must learn about group dynamics, teamwork and the acquisition and integration of new knowledge. Rather than depending exclusively upon on-the-job training, several courses have been designed to directly prepare the student for the industrial experience.

One of these is a course in software engineering wherein students are assembled into project teams of five members and given traditional industrial projects to complete. The students for this course were middlers (students between traditional sophomore and junior years) and have limited programming experience although all have taken at least one programming course (generally C++ and often Visual Basic). The limited experience forces the team to rely more heavily on the more technically astute members and is typical of what might be found in an industrial setting where a project that is outside the group's technical comfort zone may have to completed in a short period of time. The entire program is based on an industrial model and has three converging goals:

1. create a team-building environment
2. develop a unique prototype using the team
3. teach the principles of good software engineering practices

Creating a Team-Building Environment:

The individual teams were chosen at random. In an industrial setting, team members generally do not have the opportunity to choose the members of their teams and the random selection process

used in the class better simulates what happens in industry and eliminates having friends working with friends. Each team had to organize itself as a company with names, logos, mission statements and the like. Each team was charged with the responsibility of choosing appropriate roles and matching titles for each of the members based on the needs of their projects. Titles were not mandated by the instructor. It was the responsibility of the team to be sure that they chose roles properly to insure success.

Team building required that each team analyze the roles of the individuals carefully. This forced each team to take a broad look at overall project requirements and individual capabilities to compensate for any inadequacies within the group. The guidelines for this task were based on descriptions of what parts of the project needed to be managed/created and making sure that there was someone on the team charged with responsibility for each part. To react to the changing environment that occurred as the projects developed, many of the teams had to change individual roles to support the actual efforts of the team as a whole. This kind of role/responsibility change turned out to be an effective lesson in planning.

The team building experience was enhanced during weekly meetings during which many issues were raised by the "client." These issues exposed the fact that the teams had not assembled/assigned resources to deal with certain issues. These will be discussed later.

To make the team building experience more authentic, no member of a team could be fired. Although the poor performers were always the subject of discussion during office hours, the team was told that "their company" had invested too much time and money into hiring and could not afford the added expense of a new hire. The team was remanded to fixing their own problems. This challenge was often the most difficult part of the entire project and the most valuable part of the learning experience of this course.

Developing Prototypes: Let The Projects Begin!

Each team had to choose a project from a list. The following is a typical list:

ATM Machine
Public Transportation Quickest Route Finder
Courier Service Automation System
Small Airline Reservation System
Fast Food Automation System
Bookstore Inventory Control System
Building Elevator Optimizer

Teams either selected or were assigned projects from the list. Prior knowledge of any of the topics was not considered in the selection process since this more effectively represents the changing environment that would be expected in an industrial setting. The teams were told that before the end of the class, a formal presentation to "client company management" (a.k.a. the faculty) would be required. At that time, they would have to demonstrate three things:

1. An understanding of the technology/market that they were serving

2. An articulation of the specific needs of their customer
3. A working prototype

To meet the goals of the project, each team was expected to research their appropriate market area to develop some credibility with their customer. Although there were complaints from the students about lack of direction, these complaints were answered by pointing out in the industrial world companies often have to reinvent themselves and delve into new unknown market areas to survive and that their plight was no different.

Interestingly enough, each team did remarkably well in figuring out what they needed to learn and going about learning it. They involved recognized vendors, the Internet and Internet newsgroups to collect an impressive array of material.

Developing a prototype was facilitated by the myriad of Rapid Application Development tools available. Students chose from Visual C+, Visual Basic, Lotus Notes, Access and several other commercially available tools. Few guidelines were given. The only caveat was that there had to be enough of a prototype to convince the client that there was a firm understanding of the goal and that the prototype demonstrated a reasonable amount of function.

Classwork:

Each project had to follow the classic guidelines and practices taught in the software engineering course, the details of which will not be repeated here. In the classroom, the course kept pace with project development activities and outlined good software engineering practices. These included the creation of a detailed requirements document (and those teams that failed to do this suffered the consequences), good requirements planning, effective modeling, development plans, prototyping models and the like. Because coursework includes fairly standardized material, the information will not be dealt with in this paper.

Getting Started:

Little or no guidance was supplied to the teams at the beginning of the project. Rather, the overall goal was discussed in general terms so that the students had a good understanding of what was expected. In an industrial setting, there probably would have been a more structured approach. It was felt that the students should be left to themselves to allow natural leadership to surface. For some of the slow starters, questions raised during weekly meetings (“Who’s in charge here?, Who do I call for . . ., Why did you pick this language and does everyone here agree that it was a good choice, etc.”) forced the teams to create a structure and hierarchy.

Each team was asked for credentials early on. If they did not present themselves in an organized fashion, the “company representative” expressed doubt and concern and asked for references, unreasonable deadlines, working prototypes and a myriad of other time-wasting details. This forced each team to learn how to function uniformly and present a unified image. For those who did not, the first few “client meetings” were brutal.

For guidance, the students were told to rely on their limited industrial experience and common sense. The students were supplied with virtually no information about their projects. When asked for advice or direction, the Client company (the Professor) conveniently pleaded ignorance and expressed great hope that each of the teams would use their "expertise" to answer their own questions. The teams chose their own president, technical officer, marketing and sales executive and technical personnel. To add a dimension of reality, each team was assigned a controller—a student that we borrowed from the business school—to demand and enforce cost control guidelines.

Weekly Meetings—A Horror Show

In addition to conventional classroom activities, the teams held weekly meetings with their client. The model here was unusual in an academic sense yet realistic in an industrial one. The scenario used the professor as a representative from the client organization who was tasked with managing the student team. The representative acted only mildly interested in the project and claimed a nearly total lack of understanding and certainly no compassion for the team. Acting as sort of a "weak person," the client often failed to produce requested information, documents and came up with few or limited answers to questions.

The students quickly learned (often by consulting with members of other teams) that the best way to manage the client relationship was to establish firm guidelines in the form of a well organized project requirements document. Those that failed to do so typically found that requirements for the project changed randomly from meeting to meeting and issues about operational requirements platform, cost, interface changed often and without warning.

Each week, the client would (deliberately) try to change the requirements or specifications forcing the students to rely more heavily on creating a firm software requirements document. They soon learned that if specifications and requirements were not solidified in writing, the client would run them in circles and never reach resolution. They also learned the value of time because each time the client would change requirements, they would have to regroup their efforts. They eventually learned to politely ask the client to agree to specifications and requirements in writing and also got the client to agree that further changes would increase costs and delay delivery. Each time, the client reluctantly agreed to these stipulations and it was only after this agreement that real progress was made.

During the weekly meetings the client would review progress, ask questions, discuss requirements and provide limited feedback and response to questions. Most students felt that this was a difficult part of the course for a number of reasons. The client always revealed less information than requested. Excuses were always offered ("My boss doesn't want anyone to have that information. I can't find the details. Joe is on vacation and he's the only one who knows, etc.") and this forced the student teams to make their own decisions about many aspects of the project.

Many complained about this apparent lack of direction. Their comparison to previous projects where everything was defined before they began taught them a valuable lesson: Many times, an

industrial clients hire an outside organization to perform work because the client genuinely does not know either the right answers or even the right questions to ask. This approach forced the students to collaborate and investigate other potential solutions before making their proposals to the client.

The Real Challenge and the Real Lesson:

In the academic world, one would have expected that the greatest challenge to the teams would have been that of learning about a new market or technology in a short period of time. In reality, the students did quite well in meeting this challenge. In fact, even with limited technical experience (more advanced coursework is scheduled for their junior and senior years), the teams did remarkably well. The greatest challenge that they faced was that of team management.

Without exception, the teams each experienced situations in which at least one team member did not meet expectations. Each team tried varied approaches to circumvent the challenge and each experienced varying results. This was indeed the greatest lesson learned. And it certainly parallels the industrial world where there always seems to be a stumbling block. Whether dealing with an underperforming member or trying to work around idiosyncrasies of management or company culture, the greatest challenges facing technical organizations are often human ones. The teams had mixed results in dealing with these personnel issues and although the course will be revised to at least include discussions of these problems, perhaps we can do better service to our students by adding a management course to the coursework.

Conclusions:

Preparing engineering technology students for the real world is not a simple task. Technical skill preparation is something that is easily grasped. In the final analysis, each of our project teams was able to master their tasks and each provided powerful convincing demonstrations of a working prototype. As educators, we can certainly take pride in the sense of realism that we have instilled in our students.

But if we are to truly prepare our students for the real world, the area of team management and individual motivation within the team must be addressed. It was certainly the greatest challenge that these students faced in the academic world and will no doubt be a problem for them in an actual industrial setting.