

A Three-Way Stakeholder Structure for Software Engineering Course Projects

J. Scott Hawker
Computer Science Department
University of Alabama
hawker@cs.ua.edu

Abstract

Projects in a software engineering course should give students experience in the dynamics and interplay of stakeholder roles in software development in industry. We present a project structure designed around the interplay of developers, managers, and customers. We describe how we designed and implemented the project structure, and we share observations on the effectiveness of the structure. We found that the structure helped expose students to the three-way interplay of the stakeholder roles, but we also found that making the experience realistic also made it difficult to assess individual performance in the teams and assign fair grades – an important concern to students.

1. Introduction

In project-oriented software engineering courses, it is important to provide students with a project environment in which they can learn and practice skills they will need in industry. In particular, students need to experience working in teams to accommodate the needs of multiple stakeholder interests, including those of the customer and manager, as well as the developer. In this paper, we present a way to structure teams and conduct projects that provide this experience. The key concept is a three-way organization of teams in which each team works on three concurrent projects. In one project, the team acts as developers, delivering a product to another team acting as customers, under the management of a third team. In the second project for that team, they act as customers. In the third project, they act as managers. Figure 1 shows the three stakeholder roles in a project, and Table 1 shows the assignments of roles to teams in the three projects.

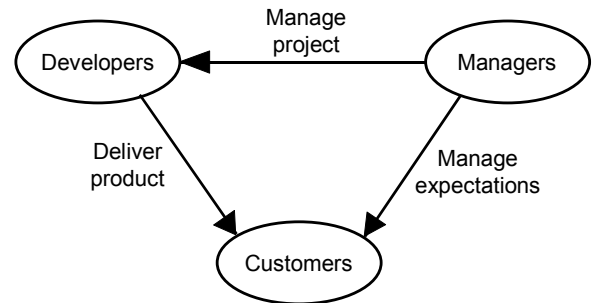


Figure 1. Three team roles in a project

Table 1. Role assignments for the three projects

Group A	Team A-1	Team A-2	Team A-3
Project A-1	Customer	Developer	Manager
Project A-2	Developer	Manager	Customer
Project A-3	Manager	Customer	Developer

In the remainder of this paper, we first describe the objectives of the software engineering course, in general, and of the projects, in particular. We next describe the details of project organization and execution. We then share some of the successes and difficulties we experienced.

2. Course description

The software engineering course is a one-semester junior/senior-level course. It is a required course for Computer Science (CS) and Computer Engineering (CE) majors, and it is an elective course for Management Information Systems (MIS) majors. We also use the project structure in a graduate CS core (required) software engineering course. We have used the project structure four times in the undergraduate course and three times in the graduate course.

The objectives of the course are built around the “Four P” characterization of software projects [1], illustrated in Figure 2. The course emphasizes the life-

cycle processes, people and team skills in projects, and techniques for the engineering of product requirements and designs. The course does not provide depth in topics of detailed design mechanisms and technologies, nor implementation and test activities, though these are certainly topics the students must address in their projects. We assume that technology and implementation details are the focus of other courses in the curriculum, so we focus more on system-level issues, the software engineering process, and team-based projects.

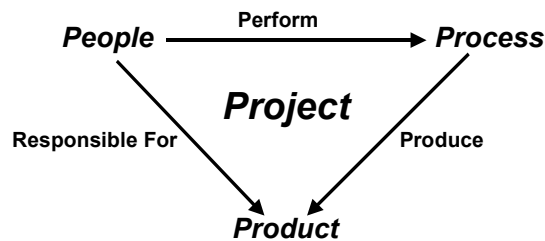


Figure 2. Four Ps of software projects: people perform engineering processes to develop and deliver products

One of the challenges in this one-semester course is providing the students with knowledge before they need it in the projects. To execute their projects, they need team skills, process knowledge, and skills for capturing and analyzing product requirements and designing, implementing and testing solutions to those requirements. In a one-semester course, we had to teach them the skills at the same time they were using them. This “just-in-time” approach does not leave much time to practice and absorb the skills.

2.1. Student background and skills

The students in all three majors have a common background in introductory computer science (two semesters), C++, and data structures. Many of the students have additional technology and management courses, bringing a diversity of skills and experience to the project teams. The CS and CE students tend to be stronger in programming-level topics such as languages, data structures, and algorithms, but weak in system modeling, analysis, and architecture design and in software engineering processes. The MIS students tend to be stronger in system-level analysis and the software engineering processes, but weak in programming-level topics. In the graduate course, there is even more strength in technical skills, but the process and team skills are also weak. The graduate course has the added diversity of numerous international students.

All the students are weak in problem-solving skills and dealing with open-ended problems (a common problem in undergraduate curricula). They are used to

being told what to do, how to do it, what to turn in, and when they are done. They are uncomfortable when these are not clearly laid out in their projects, and they become especially uncomfortable when this results in uncertainty on how to obtain a good grade in the course.

Similarly, the students are weak in skills for working on fairly-large projects. They struggle to break up the work into increments and into tasks that can be carried out concurrently by team members. They have difficulty setting deadlines and managing their effort to those deadlines, and in identifying and managing risk. The projects are large enough that the students must (or, at least, should) work together as members of a team. They discover that they are weak at building a diverse collection of skills and personal goals into a balanced team with common goals, and they are weak at addressing the conflicts that arise when the team does not function well. Again, grades become a driving concern as individuals worry about how they are to obtain a good grade “in spite of” their teammates, rather than with the help of their teammates.

2.2. Course project objectives

The course projects are designed to address the issues of students building and managing teams addressing open-ended problems, and to expose the students to the dynamics of multiple stakeholder roles in a project. The projects address teamwork, applying technical and team skills to solve real-world problems, the interaction of developers, customers, and managers, and the use of models to organize and communicate about work. The course and project structure do not avoid or resolve the issues, since these are issues the students will experience in industry. Rather, the course and project structure attempt to give the students some basic skills and a setting in which they can experience these issues and develop their skills and comfort level when facing issues. The objective is to provide a simulation of the real world of software engineering that has enough fidelity to be realistic but is provided in a setting that is safe and conducive to learning.

We based the design of the project structure on our industry experience in developing large-scale software. Unfortunately, we did not also base our design on reported research and experience in teaching team project work. As we will describe later in this paper, our striving for real-world experiences resulted in the students feeling unsafe, especially regarding grades and exposure to interpersonal conflict. The students may have learned, but the lessons may have been harder than necessary. Before discussing these observations, we first describe the project structure in more detail.

3. Project structure and execution

The project structure design includes the following topics: team roles, project selection and launch, project reviews, peer performance appraisals, and project deliverables and grades. This section presents details on these topics.

3.1. Roles

Figure 1 illustrates the basic structure of the teams as stakeholder roles interacting in a project. There were three to five people on each team, and two to four groups of three teams, depending on class size. The students received descriptions and expectations of each of the roles and their interaction.

3.2. Project selection and launch

The project teams selected their own product to deliver. They were encouraged to create a tool or utility they would want to use, or to select a product that would give them exposure to a new technology or application domain. Our hope was that, by selecting a product they were interested in, they would be motivated to work on the project. In theory, the customers were to be responsible for specifying product requirements and the management teams were to align requirement priorities with team skills and available time. In practice, the development team had the most control over product selection.

The project teams (in particular, the development teams) also selected the technology appropriate for implementing the product. This gave them experience in selecting among design alternatives. It also allowed them to match technology choices to team skills and to desires to learn new technologies. The students were reminded that the focus of the course was more on process than on product technology and implementation. They were encouraged to select a relatively small, low technical risk product so they would have time to address team and process issues.

The process model to use was a choice that the managers and developers made. Since the students had little, if any, prior exposure to alternate process models (such as waterfall, iterative, etc. [2]), they were encouraged to perform up-front prototyping of user interfaces to validate requirements and prototyping of product technologies where technology was a risk. After that, they were encouraged to follow a phased analysis, design, implementation, and test effort, but were also encouraged to look for core functionality that they could develop in an early increment and additional functionality they could develop in a subsequent increment.

To get the projects moving and addressing the breadth of project issues, the teams spent the first two weeks in a concept formulation phase. In this phase, they defined their project from the three stakeholder perspectives.

- For the customer perspective, they defined a product concept, including product features and motivation ("business drivers") for choosing that product.
- For the developer perspective, they defined a technical approach, including identification of key subsystems and choices of programming languages, infrastructure technologies (operating systems, database management systems, communications, etc.) and development tools and equipment.
- For the manager perspective, they defined a project management plan, including schedule, increments, risk management, and developer training.

At the end of this project concept formulation period, there was an in-class review of the project concepts. This helped the teams identify issues and opportunities to steer their projects toward success, and it allowed them to learn about each other's projects.

The students also received a one-hour lecture and in-class exercises on team skills, including topics on leveraging diversity, decision-making, problem-solving, meeting management, communications, conflict management, and steps in building and evolving a team.

3.3. Project reviews

There were a series of project reviews during the semester. The first project review was the concept review, the second review focused on product design (but also had a management status portion), the third review was to assess progress and barriers to completion, and the fourth review focused on the final assessment of the product and process. The combined three-way teams developed artifacts for review and materials for the in-class portion of the review.

The teams were instructed that the purpose of the reviews was to obtain peer feedback on their project with the goal of improving the quality of the developed product and of the development process. The review was to have two aspects: a management portion focusing on project progress, risks, and process effectiveness; and a technical portion focusing on system analysis, design, implementation, and test decisions and issues.

The three-way teams were to work cooperatively to develop review materials. The management team typically developed the management portion, and the development team developed the technical portion. Approximately fifteen minutes in class were available for each project's review.

3.4. Peer performance appraisals

Performance appraisal is a necessary, but often uncomfortable and difficult, aspect of the life of a software engineer in industry. Beyond giving the students experience in individual performance appraisals, the course appraisals were essential to obtaining information on which to base individual grades in the project efforts. The appraisals were also important in identifying issues that needed to be addressed beyond the team boundaries.

Peer performance appraisals occurred during the course exams. As part of the exam, the students filled out a form stating whether each team member exceeded, met, or did not meet expectations in the following categories:

- Teamwork
- Knowledge/Skills
- Dependability
- Initiative and Creativity
- Adaptability and Flexibility
- Delivery of results

Manager assessment of developer performance was an important item in this assessment, as it was one of the ways managers could exert influence over their developers.

Every two to three weeks, the students, in their developer role, filled out a team progress and effectiveness report where they assessed whether they were meeting goals and functioning effectively, whether their managers and customers were functioning effectively, and whether specific team members were doing more or less than their share of work. Team progress and effectiveness appraisals helped the team pause long enough to identify successes to celebrate and issues to address, and also gave an additional opportunity to communicate interpersonal issues to the instructor.

3.5. Project deliverables and grades

During the semester, the students were encouraged to create a folder for all project artifacts. The folder contents is evidence that the students followed a process in their project, rather than simply cranking out code. In this folder, they were to include traditional product-oriented artifacts such as requirements documents, design models, code, and test results, as well as process-oriented artifacts such as schedules, risk assessments, review presentation materials, meeting minutes, scratch paper notes from discussions, etc. Since there were three teams working on each project (covering the three roles of manager, customer and developer), all three teams were involved in assembling the project folder. At the end of the semester, the students turned in the project folder as the project deliverable.

The project received a grade based on evidence in the folder that the students carried out a software engineering process. We used a checklist of expected artifacts with a general assessment of the quality of those artifacts – do they serve the purpose of guiding the product

development? We did not directly assess the quality of the models and designs, as that was not the focus of the course, but haphazard attempts at modeling did indicate a lack of attention to the process.

In practice, there was very little to distinguish project grades. They uniformly all received the same grade. Of course, the students did not know this, and the assessment was always made after the projects were complete.

The first few times we used the three-way project structure, we assigned the project grade to the development team, since that was where the bulk of their work went. However, we found that the managers and customers failed to perform their roles because there was no grade incentive. In response, and reflecting reality in industry, we emphasized that management was the role most accountable for project success, so we assigned 50% of the project grade to the management team, 40% to the developer team, and 10% to the customer team. This was a significant source of unease in students feeling that they were receiving fair grades. In particular, the managers felt they had little, if any, influence over the performance of the developers, yet 50% of their grade was based on the performance of the developers

To provide an element of reward and encouragement that the managers could use to influence their developers, the managers were allocated a pool of bonus points to award their developers as they saw fit. Unfortunately, these points were usually not awarded until the end of the semester, which was too late to influence developers. Sometimes, one team of developers suffered in comparison to their classmates when their managers didn't bother to award any bonus points, but other manager teams rewarded their developers. As another element of influence, we paid particular attention to management assessment of the performance of their developers, and all students were told this.

The individuals received a baseline project grade computed as above. This baseline was then adjusted up or down depending on the peer appraisals and instructor observations of daily performance and professionalism. If the peer appraisals reflected a positive or negative trend that could be corroborated from multiple sources, it influenced this individual adjustment. The students were told their individual project grade could be adjusted up or down by up to 50% and, since the projects were 30% of each student's overall course grade, individual performance could result in a full letter grade or more of adjustment. In practice, only about 25% of the students received project grade adjustments, and of those, most were 10%. In no case did an adjustment exceed 30%, and a 30% adjustment corresponded to students who clearly failed to contribute to the project.

4. Observations

In this section, we share a number of observations of the three-way teams in operation.

4.1. The reality of grades

The biggest challenge in the three-way project structure was to balance the fidelity of simulating the real-world to provide a valuable learning experience with the importance and difficulty of fair grading in team projects. The students were reluctant to engage in their team roles because they did not understand how it reflected reality and because they saw it as disconnected from their grades. When the grading structure was adjusted to better connect the roles to the grades, the result was uneasiness in the fairness of grades and even accusations of unfair grading or of a student sabotaging the grade of a classmate.

We also found that there is not enough work as a manager and a customer to spread that work across multiple team members. We found that some teams would assign the team responsibility for management to one person, the responsibility as a customer to another person, and then focus on dividing up the development work among the team members. The result was that not all team members experienced the manager and customer roles, and that significant impact on team grades was placed in control of one or two individuals.

Even though project managers are held accountable for project success in the real world, the students did not engage in this role, even when their grade began to depend on it. Even though customers can receive a product they desire by being clear and persistent on product requirements, the students did not engage as customers. Since the bulk of the students' work was as developers, this became an almost exclusive role. In some ways, this is to be expected. Most students will have a first job as a developer, not as a manager of developers or as a customer specifying product requirements to developers.

4.2. Peer appraisals

The students were reluctant to provide accurate, serious peer appraisals. Those students striving for top grades in the course tended to take on more than their share of responsibility in the projects, but the peer appraisals and progress assessments did not often report teammates doing more or less than their share. The peer appraisals were uniformly high: "my team is great!" Even when a bonus point system was used, where the students had a certain number of points ("dollars") to allocate to their teammates, they tended to give everyone an equal share. When asked to assess the performance of the

management team as managers and the customer team as customers, there was no differentiation. Everyone performed equally well in a uniformly undistinguished role.

One inter-team dynamic that sometimes appeared was a reluctance to enforce responsibility and accountability of a person or team because, in another project within the same group, that person or team had another role that they could use in revenge.

4.3. Process performance

Having distinct team responsibility for different stakeholder roles helped assure that the teams followed the basic steps of the software engineering process. Having customers encouraged the developers to clarify and validate product requirements. Management responsibilities encouraged the identification and tracking of risks and milestones. Having project reviews across teams encouraged the teams to create artifacts that explained the product or process, and it allowed the reviewers to experience forming a quick intuition on product and process quality and approaches to improvement. Having peer appraisals encouraged the students to reflect on their own behavior and performance. Having to turn in a complete project folder encouraged the teams to cover all stakeholder perspectives, instead of having a myopic focus on getting the product code to run.

4.4. Anecdotes

There were some interesting episodes where parts of the class were able to experience some "reality."

In a number of cases, a student would drop the course well into the semester. In these cases, we facilitated a discussion between the management team who lost a developer resource and the developers and customers. In most cases, the teams agreed to reduce product scope. In one case, the teams agreed to abandon the project and re-allocate the development resources to the remaining two projects in the group. In this case, the team that lost a member continued in its original roles as customer and manager to the other projects. Also in this case, the remaining project teams were able to experience the effort and opportunity to restructure an existing project to bring in a new teammate. The teams took specific steps to rebuild the team with the new member and to modify the project to incorporate the new skill set.

In one case, an individual in a customer team pushed hard on the reality of the simulation. At one of the reviews, they became very demanding that their requirements be met. At the final review, they made it clear that their requirements were not met. The responsible management team pulled out a document capturing the customer team's agreement on the minimum

acceptable requirements, which the product met. This individual challenged the skills of the management team in managing customer expectations!

There were a few cases of significant, poorly-managed interpersonal conflict. Some of the gifted students were poor team players, either treating their teammates with a level of contempt, or taking on more than their fair share of the team effort. Management was ineffective in resolving these conflicts, again because of a feeling of pride and superiority of a gifted team member. In two other cases of conflict, the development team and management team could not, or would not, agree on what approach to take on an issue. Voices and emotions were raised and accusations were made. As the semester wore on, blame and statements such as “if you would have done what I told you ...” occurred. “Senior management” (instructor) intervention helped smooth the conflict, but the conflict still lingered. In these cases of conflict, peer appraisals and instructor observation made it clear which team members were irresponsible and which behaved professionally.

We found that the project structure was more successful in the graduate courses than the undergraduate courses. The graduate students were more conscientious about carrying out their responsibilities, and they seemed less concerned about grades and more concerned about learning than the undergraduate students.

5. Conclusion

In many respects, the three-way project structure was successful. It exposed the students to the roles and interaction dynamics between customer, manager, and developer stakeholders in a software development project. It engaged them in some of the difficult aspects of real-world projects, such as peer appraisals and being held accountable as stakeholders for things where there is little direct control but where team and interpersonal skills can have significant impact. The three-way project structure encouraged them to look at the development projects as more than creating code that compiles and runs. The projects may not have engaged the students in their roles at the same level they would experience in industry, but it did provide them significant exposure to the intention and interplay of the roles as well as some experience in seeing the roles in action.

Unfortunately, grades are a primary motivation for students to engage in a course. Learning objectives often become a distant second to obtaining grades. It is difficult to balance the inter-dependent nature of multi-role, multi-member team projects with the need to provide fair individual grades at the end of the project. This resulted in the students having a reluctance to engage in the “simulated” project roles, thereby reducing the learning effectiveness of the three-way project structure.

The importance of grades over simulated reality was the primary reason that we have abandoned the three-way project structure in our current course structure. We have preserved some aspects, such as peer reviews of product and process artifacts and peer performance appraisals. We have concluded that our way of creating management and customer roles was too artificial in a classroom setting and had too much impact on our ability to provide fair grades.

We are now searching the literature and speaking with our colleagues to find ways to modify the three-way project structure. We believe there is value to exposing the students to the dynamics of the three stakeholder roles, so we are looking for better ways to assess individual performance in assigning fair grades, while still emphasizing the inter-dependence and dynamics of software development teams in the real world outside the classroom.

6. Acknowledgements

Co-teaching the software engineering course with Marcus Brown provided insight into the effectiveness of the project structure and the relationship between the project aspects of the course and other aspects. Discussions with Russ Pimmel and other colleagues provided insight into teaching team skills and assessing individual performance in teams.

7. References

- [1] Jacobson, I., G. Booch, and J. Rumbaugh, *The Unified Software Development Process*, Addison-Wesley, Reading, MA, 1998, pp. 15-31.
- [2] Sommerville, I, *Software Engineering* (6th edition), Addison-Wesley, Harlow, England, 2001.