CS Capstone: Lessons from the Field

Robert Adams and Jamal Alsabbagh
School of Computing and Information Systems
Grand Valley State University
adams@cis.gvsu.edu and alsabbaj@gvsu.edu

During the several years that we have been teaching the computer science capstone course we have learned many valuable lessons. This paper describes the pedagogy of our course along three axes: product, process, and professionalism. The paper illustrates how our course ensures that the student, by demonstrating each of these elements, will be a competent professional upon graduation. Finally, the paper concludes with other lessons learned regarding motivation, team-formation, individual student evaluations, and development methodology. Our goal is to provide ideas for others engaged in capstone education, as well as to elicit feedback from that same audience.

Introduction

A recent column in the SIGCSE Bulletin[1] described three overarching goals of computer science capstone courses, and named them the three "P"s. They are product, process, and progression in learning. "Product" refers to the actual software being produced by CS students in their capstone course. We all care that our student can produce first-rate software. "Process" refers to ensuring that our students follow an established software development methodology, and do not approach their projects in an ad hoc manner. Furthermore, we expand the goal to include communication and presentation skills. "Progression in learning" refers to an observable increase in the breadth and depth of student learning throughout their time as undergraduate students.

It can hardly be argued that these three goals are (or should be) present in every CS capstone experience. However, it seems that progression in learning is an issue larger than the capstone course itself. For example, Clear[1] describes the use of student portfolios to assess progression in learning throughout a student's career. This is an issue larger than the capstone course. Therefore, we offer an alternative third "P": professionalism. In the context of a CS capstone course, professionalism refers to personal behaviors exhibited by the student[2]. We want students to make informed ethical decisions within the context of a software development project.

This paper addresses how the CS capstone course at GVSU addresses each of these issues. After describing what is currently happening in our CS capstone course, we provide several observations and lessons learned. Our goal is to provide ideas for others engaged in capstone education, as well as to elicit feedback from that same audience. Our hope, like others that teach CS capstone courses, is to improve the quality of the capstone experience in order to produce the best CS graduates.

The next few sections describe the pedagogy of our CS capstone course. The final section contains our reflections on teaching the course for the past several years.

Product

The "product" dimension of the capstone course deals with the quality of the projects being produced by students. This is a fairly easy goal to justify - we all want our CS graduates to be able to produce efficient yet elegant code. The difficult task is how to judge the quality of the projects.

Motivational Factors

For the majority of our students, the capstone project is their most significant semester-long project during their degree program. Our first step in promoting quality is motivational, where students are reminded at the beginning of the semester that their capstone project and experience will probably be the most significant part of their professional portfolio upon graduation. Furthermore, students are strongly encouraged to showcase their work publicly in venues such as "Project Day" and "Student Scholarship Day" that are organized, respectively, by the College of Engineering and Computing and the University.

Although hard to quantify, we believe the motivational element of a public presentation and demo help motivate teams to produce quality final products. Project Day and Student Scholarship Day involve faculty and students from other departments and the general public. We believe the added pressure of a wider audience helps to motivate students to do their best.

Our second way of promoting quality is by allowing individual teams to identify their own project ideas. Our approach was motivated by the observation that when we assigned the projects in the past, students treated them as very large assignments and perceived their role as one of meeting the specs. In contrast, when they are allowed to come up with projects that they wanted to pursue, students have a vested sense of ownership of the project. Students seem to be significantly more motivated, pursue independent learning, and exhibit a sense of ownership and pride. Projects are not considered "work" that instructors make them do. Rather, successful completion of the project is often seen as personal fulfillment.

Deliverables

As expected, motivation does not, by itself, ensure quality. In order to monitor progress and quality, each team is required to submit a prospectus, a design document, a final report, and a set of periodic status reports. The content and deadlines during the 15-week semester for the deliverables are discussed next.

The prospectus is due during the 3rd week and must contain, at least, the following items: (1) a clear description of the project scope; (2) a short description of the candidate tools, languages, or technologies that will be used; (3) a breakdown of the responsibilities of each team member; and (4) tentative timeline at the subtask level. The instructor reviews the prospectus in order to ensure that the type of project and its scope is appropriate for the capstone course. More specifically, we expect capstone projects to include topics from several sub-disciplines within computer science, e.g., networking, artificial intelligence, graphics, databases, programming languages, parallel algorithms, etc.

The design document is due during the 5th week. This document describes how the team will organize the project, what development methodology will be followed, and the tools and libraries they will use. The design document forces students to think about the overall structure of the project, to reflect upon an appropriate development methodology, and to research possible implementation technologies. The instructor reviews the design document to ensure that the project involves sufficient components to be considered a capstone experience. For example, a project that involved nothing more than tools learned in regular classes would not be acceptable. Rather, the instructor looks for projects that require a fair amount of independent learning of tools, techniques, and technologies. In fact, over the last several years all teams used languages, tools, and technologies that were not covered directly in any formal course.

The final report is due during the 15th week. It is a formal report that documents the product and the process. In addition to the technical contents, the report must also contain a section on professional ethics and will be discussed in more detail in the Professionalism section below.

In addition to the above three major deliverables, each team is required to submit three status reports during the 7th, 10th, and 13th week. The purpose of the status reports is to both monitor progress and remind students of the importance of continually making progress on their capstone project. The status reports require students to describe what they wanted to accomplish (from their design document), what they really accomplished, and to reflect on their progress to date.

Finally, each team is required to give a 30-minute presentation and demo of its project during the 15th week. More information on the role of the presentation is given in the section on Professionalism below.

While the above deliverables are team-prepared, the instructor also requires every student to submit a periodic assessment of each member in his/her team. The contents and purpose of this assessment is also discussed below in the Professionalism section.

Evaluation

The combination of motivation factors and deliverables ensures that our students are technically strong. By the end of the term they have successfully built a large, complex piece of software without close instructor intervention.

However, evaluating individual effort in a team setting is difficult. The weekly journals help in this regard, but on more than one occasion teams have approached the instructor to say that one of their team member's journal was incorrect, i.e., the student was making up journal entries. Therefore, in our next course offering we will be mandating the use of Subversion[4] in order to be able to track changes made by individual students[5].

Process

The "process" aspect of the capstone course addresses the fact that we want our students to learn how to build software of significant size. Ideally, students should understand the strengths and weaknesses of several software development methodologies, and be able to choose a methodology to manage their projects.

In our capstone course, the instructor gives students the freedom to choose a methodology that they describe in their design document. The instructor often hints that students choose a methodology that they haven't used before. The rationale is that if students choose an already familiar methodology, they are not gaining the broader development experience that we would like them to have before they graduate.

Students are required to reflect upon their chosen methodology in their final report and presentation. The instructor asks them to critique their methodology and discuss its advantages and disadvantages. The instructor does not grade the process/methodology itself. Rather, the instructor looks to see that each student thoughtfully discusses their chosen methodology in the context of their project.

Evaluation

In our CS program, the capstone course is the only place where students get a semester-long project in which they must identify and follow a development methodology into practice. Most of our other courses either have shorter projects, or do not require any particular methodology be followed. Our rationale is that by the end of the capstone course, students have had a significant immersion into a methodology of their choosing. Furthermore, their required reflection on the strengths and weaknesses of their chosen methodology enable students to critically evaluate a methodology on their own.

It is interesting to note that in the many years we have allowed students the freedom to choose, only one group has chosen the classic waterfall model. Every other team has chosen an agile methodology. Perhaps this is due to the negative spin put on the waterfall model by the more vocal agile community, or perhaps they do not feel they have enough time for the waterfall model to be successful. However, this is mere speculation, and an avenue for more investigation in the future.

Professionalism

The final aspect addressed by our capstone course is "professionalism". In our context, professionalism refers to the behaviors exhibited by students during the course. During the capstone course, the instructor attempts to answer the question "does the student act professionally?"

Unfortunately, professionalism is a loaded term and can have several meanings. We have decided to limit ourselves to the following. Being "professional" means that one

- is a valuable, contributing member of a team.
- does the best work one is capable of.
- understands how a code of ethics affects one's discipline.
- exhibits a command of oral presentation skills.

Measuring Teamwork

Assessing a student's role in a team is accomplished through periodic peer evaluations. Every 2-3 weeks every student is required to submit an evaluation of his team members. The evaluations require commenting on whether the other team members made good technical contributions, were responsible and dependable, and showed initiative.

The evaluations serve two purposes. First, the fact that every student is evaluated by their peers reminds them of the importance of professional conduct. Second, the periodic reports alert the course mentor as early as possible to problems that may require intervention. Depending on the seriousness of the comments the course mentor may choose to talk with the student (and perhaps the entire team) to try to correct any issues within the team. In the extreme case a student may be removed from a team and fail the course.

Measuring Workmanship

Quality of work is measured in several ways. All the written deliverables are graded on clarity, organization, and quality. The final presentation is graded on how well it is delivered (e.g., was it rehearsed?). The team's final demonstration is graded on how well it is presented, and whether or not it demonstrates the relevant features of the final product.

Since every team chooses its own project, the quality of work is not graded exclusively on the length of the feature list. It is virtually impossible to compare two different projects in order to decide which one is "better". Therefore, each project is judged on its own merits based upon the initial project prospectus, along with the periodic milestone reports. Our expectation is that while students will do their best, they probably will not complete every feature they hoped for.

Measuring Ethics

Students are required to read, understand, and reflect upon the Software Engineering Code of Ethics and Professional Practice (SECEPP)[3]. We discuss the paper at the beginning of the semester and tell students that they are expected to apply its relevant principles throughout the semester. In particular, the instructor requires them to focus on Principle 3 of the SECEPP that deals with product development and to include in their final report a section in which they must identify all items under principle 3 that are relevant to their project. For each such item, the report must include a clear comment on how the team addressed that item during the development of the project.

Presentation Skills

Each team is required to give a 30-minute presentation and demo of its project during the 15th week. The presentation ensures that students are given an opportunity to describe their project to an audience that is not familiar with it.

As an alternative to an in-class presentation, teams are encouraged to present at venues that have a general university-wide audience (as described above). This often is a significant challenge for students that are used to presenting only to their CS friends. Last semester three (out of five groups) were proud of their work and excited enough to display it at the university-wide Student Scholarship Day and as wells as the college-wide project day, even though participation was completely voluntary.

Evaluation

All four measures of professionalism contribute to the student's final grade. If the peer evaluations are poor or if a student's contribution to the final presentation is inadequate, then the student may receive a grade lower than the rest of the team.

The periodic team evaluations are submitted anonymously, and students tend to evaluate each other honestly (student evaluation of each other tends to match the instructor's own evaluation of each student). Furthermore, students are encouraged to contact the instructor if a team member is not contributing fully to the project. So far, the authors have not had incidents of an unproductive student being shielded by his team. Rather, the reverse seems to be the rule: teams want to get rid of unproductive team members as soon as possible.

Furthermore, we try to provide for better team coherence by allowing students to form their own teams. Historically the course enrolls approximately 30 students, which are divided into teams of 4-6 students. By the tie students get to the senior capstone course they have had several classes with their peers and have formed friendships with several of them. Allowing those friendships to persist throughout the capstone has meant the instructors do not have to closely monitor teams and remind them to get along. In fact, students have often approached the instructors before the term begins and asked to form teams and start work.

The team evaluations also help in measuring workmanship. Students are quick to note if another student submits code that doesn't work, or doesn't work well with existing code. The team evaluations, along with the final demo, alleviate having the instructor look closely through the code itself.

Finally, the students' ethical reflections have been surprisingly good. Our students clearly appreciate the role of a professional code of ethics, as it applies to software development. Their reflection documents demonstrate their knowledge of the SECEPP.

Conclusions

Teaching any capstone course is challenging. One must balance many factors in order to make it a successful course, both from the perspective of the students, as well as the perspective of the university. Framing those challenges within the contexts of "product", "process", and "professionalism" allows one to focus on the salient components of the capstone course, ensuring a successful student experience.

We have received several comments from students over the many years we have been teaching the capstone course. Typical comments include "allowing us to choose our own projects…is always more enjoyable" and "it is great that the class is split up into milestones, such that the entire grade isn't earned on the last day of class but rather it is built up from the beginning of the semester through to the end".

In summary, we believe our students have a foundation for understanding and practicing professional behavior. They know how to work together in a team, they know that shoddy workmanship is not tolerated, they are aware of how a code of ethics helps to answer possible dilemmas during project development, and they understand the challenges of presenting their information to a wide audience.

References

1. Clear, T. 2009. Thinking ISsues: the three p's of capstone project performance. SIGCSE Bull. 41, 2 (Jun. 2009), 69-70. DOI= http://doi.acm.org.ezproxy.gvsu.edu/10.1145/1595453.1595468
2. ACM Code of Ethics and Professional Conduct. http://www.acm.org/about/code-of-ethics. Retrieved on 12/15/09.
3. D. Gotterbarn, K. Miller, and S. Rogerson, Software Engineering Code of Ethics is Approved, Communications of the ACM, Vol. 42, No. 10 (October 1999), 102 107
4. Subversion is available at http://subversion.tigris.org/
5. Jones, C. 2010. Using subversion as an aid in evaluating individuals working on a group coding project. J. Comput. Small Coll. 25, 3 (Jan. 2010), 18-23.