Teaching Design Thinking, Writing, and Oral Presentation: Lessons Learned from the Computer Science Senior Design Course at GW

Gabriel Parmer, Rahul Simha, Chris Toombs, Poorvi Vora & Timothy Wood
Department of Computer Science
The George Washington University
Washington DC 20052
{gparmer,simha,cctoombs,poorvi,timwood} @ gwu.edu

Abstract
Computer science students in the B.S. program at George Washington University take an 8-credit one-year course sequence in senior design during which students must demonstrate working software containing a significant algorithmic component. The course features many elements including: design and software engineering, writing for broad audiences, oral presentations, staged development of the student product, use of modern software tools, and contact with alumni to bridge students towards their future work environments. Two types of data have shaped the lessons learned: formal focus groups conducted with each class of senior design students, and informal feedback from well-meaning alumni. The interesting conclusion is that the very features seniors tend to complain about – design, writing and oral presentation – are the ones alumni report as the most valuable.

1.0 Introduction

Capstone courses in American higher education are thought to date back to the 1850's, when colleges like Williams College offered capstone-like courses.[3] However, the real impetus for widespread adoption would await another century: capstones entered the common educational lexicon as formal curricula began to embrace them in the 1980s. Capstone's modern curricula are designed to let students integrate knowledge from foundational courses while also developing broader skills such as presentation, writing, teamwork, and of course, design, consistent with the goals outlined in the influential Green Report from the ASEE.[2] There are now even textbooks that explain in detail the possible micro-structure of capstone courses in engineering.[4] A recent paper (2012) provides a survey of the literature on capstone courses.[1]

The computer science capstone (senior design) course has evolved in direct succession of engineering capstones offered at George Washington University beginning in 1963, in the then electrical engineering department. When the undergraduate computer science (CS) program was launched, the CS capstone evolved into a form maintained through the transition (in 1999) into a separate computer science department until 2010 by one key instructor who has since retired. This paper's authors modified and refined the CS capstone into its current form. The relatively

small scale of our department (typically 20-40 students per graduating class) has enabled us to build a senior design course that focuses on technical design, algorithmic depth, and presentation skills, which we believe are crucial for producing successful and engaged alumni.

In the current version of the capstone, computer science students in the GW B.S. program take an 8-credit one-year course sequence in senior design for their course requirements. In this course, students must demonstrate working software containing a significant algorithmic component developed by the student, and for an application that is new to the world. The course features many elements including: design and software engineering, writing for broad audiences, oral presentations, staged development of the student product, use of modern software tools, and contact with alumni to bridge students towards their future work environments. The lessons learned during several years of experimenting with these various features form the core of this paper. Two types of data have shaped these lessons: formal focus groups conducted with each class of senior design students, and informal feedback from well-meaning alumni. The interesting conclusion is that the very features seniors tend to complain about – design, writing and oral presentation – are the ones alumni report as the most valuable.

The paper and ASEE talk will begin with the course objectives and reasons for overhauling the course (Section 2). We then, in Section 3, describe the elements of the course, focusing in greater detail on the three main elements: design, writing and oral presentation. Finally, we present lessons learned in Section 4.

2.0 Course Objectives and Evolution

The 8 credits of senior design are associated with a year long Fall-Spring sequence of courses, each carrying 4 credits. The course has core junior-level courses as prerequisites so that students enter with some substantive computer science fundamentals. We list below the formal course objectives. In the two courses, students will:

1. Learn key elements in the development of a significant year-long computer science project: planning, specification, design, analysis, and implementation.
2. Apply concepts from software engineering to the project: requirements, specification, reuse, documentation, verification and validation, testing, configuration management.
3. Learn to write about and practice presentations about important aspects of the project: the case for launching the project, status reports, design, and implementation plan.
4. Demonstrate a working project.
5. Demonstrate an understanding of how knowledge and skill in computer science courses played a role in the project.
6. Explore issues related to local and global impact of computing, as well as social impact issues.
7. Experience working collaboratively.

As such, the course objectives are in alignment with typical ABET-accredited senior design courses. However, several factors have played into how the objectives are met in the course. In particular, years of exit surveys revealed issues in the previous design of the course that we needed to address. We list some of these here:

- ***Software engineering***. Prior to 2010, students took a junior-level course in software engineering prior to senior design. Students complained both that they did not learn much from that course because it was too abstract and accompanied by a poorly illustrative toy project, and that none of it was remembered when they needed it most in senior design. As a result, we decided to eliminate the formal software engineering course and instead integrate the teaching of software engineering directly into senior design using the students' own projects as driving examples.
- ***Scaffolded introduction to design***. Another element of the pre-2010 course that bothered students was that they were required to spend the entire first semester only on design, and then had to implement in the second semester, often without an opportunity to revisit the design. Recognizing that students have difficulty with design when they know too little about what their project entails, we have instituted a scaffolded approach: students first, in a bootcamp phase, explore their project through a little coding and implementation before committing to a design. The design process itself has several elements spread over the first semester as the students write more code. These include high-level user-interface design, the design of main components, algorithmic design, and testing.
- ***Algorithmic component***. While originally students had great flexibility to propose ideas for projects, we have since imposed some constraints on the type of project. We did not want students to be doing lots of incremental busy work, as for example in building a large website. Neither did we want student projects to be about crafting a video game or animation, or solely conducting research. In sum, we require students to build an actual product that is justifiably different from existing products, as one would expect from a startup, but also one which contains a core algorithmic challenge that is addressed by the student. Thus, a student could in fact build a website as long as the interaction with the website resulted in a serious algorithmic component in the back end, as for example, with a website that performs a useful computation.
- ***Presentation and writing***. While presentation and writing requirements existed prior to 2010, we made substantial changes based on student exit surveys and on new writing-in-the-disciplines requirements imposed by the university. Writing assignments now address different audiences, and range from technical to business plans. Furthermore, they are graded by a graduate student from the university's writing program, who is charged with applying a variety of criteria designed to improve writing for a non-technical audience. Whereas earlier students complained about not knowing how to present, we now conduct an intensive all-day workshop on presentation skills at the beginning of the course, and constantly reinforce those skills through detailed feedback on multiple presentations throughout the year.

- ***Transition to the outside world***. One new goal we added was to find ways to better prepare students for their future. We have discovered that involving recent alumni in this process greatly motivates currently enrolled students. For example, alumni not only talk about their experiences, they teach some of the classes and conduct mock interviews. The students in the course love to engage with alumni and find their feedback (which is not different from that of professors) to be highly credible.
- **Teamwork**. Prior to 2010, all projects were individual projects. While contemplating teamwork, students forcefully argued against working in teams. They felt greater ownership of their own idea as an individual, and most students intensely dislike being dependent on others. Nonetheless, we wanted to introduce some elements of group learning, which we describe below. In the most recent offering (2015-16), we have begun experimenting with team projects for some students.
- **Pre-senior design**. In hoping to get students started earlier, we experimented with a 1-credit "get started with senior design" course in the Spring of the junior year. Eventually, after evidence that this approach did not have any positive impact, this aspect of senior design was dropped.

3.0 Elements of the Senior Design Sequence in Detail

3.1 Instructional team and responsibilities

Depending on enrollment, the course has 2-4 instructors and an alumus as part-time instructor. Currently, with 26 students, one professor is the lead instructor responsible for overseeing the course, communicating with students and coordinating amongst the other instructors. Two other faculty serve mainly as project mentors, but also attend student presentations and provide feedback. Finally, the alumus (Mr. Christopher Toombs, from the class of 2005) is an experienced professional with a keen interest in teaching students about design, especially from a real-world perspective. He has taken responsibility for the design lectures and design homeworks.

Alumni. The other alumni and industry professionals participate in two ways. Each delivers an interactive lecture on some aspect of the real world. For example, one of the popular lectures is on "life after graduation." Another is on modern software tools. The alumni also conduct mock interviews and attend the final presentations to give feedback to instructors. It is one way in which we get a sense of how well prepared (or not) our students are.

Mentoring. A key role for faculty is project mentoring. In the Fall semester, faculty mentors meet with their students weekly in a time set aside for the course. Students are asked to maintain a logbook with to-do lists, and expectations of progress. We also structure these meetings around groups of students to expose students to the technical details of each other's projects, so that they can help each other if one student's problems have already been addressed by a peer,

and to make the mentoring workload reasonable. We have also observed that peer pressure can play a healthy role at this stage. If even one of the students works hard on the project and has something to show, this appears to motivate the others to work harder as well. Mentors also make time during separate office hours for students with additional difficulties. Finally, each student performs their work in the open on a shared repository (such as github.com) and is required to make regular commits.

3.2 Chronology

The course is structured around three phases: (1) the bootcamp, (2) development, and (3) the "road show." The terms are loosely, but deliberately, borrowed from the world of startups where the road show refers to a mature startup just prior to going public.

Bootcamp. The bootcamp phase lasts roughly a month and enables the students to explore and learn the underlying technologies of their projects. For example, a student whose project involves drone surveillance must learn the kit that comes with the drone, and then separately, something like openCV to handle image streams, and also simple client-server paradigms and databases. This bootcamp phase is critical in several ways. First, it recognizes that for the most part students have not had substantially long projects in prior courses: the most significant is perhaps a three-week project in operating systems. Nor have they used, much less assembled, large open-source libraries to achieve some end. Thus, they have very little sense of what it takes to build their project, and therefore very little sense of how to even think about design. The bootcamp is essential in getting students to understand the different elements of their project, the project's expected complexities, and having written some preliminary code, puts them in position to think more deeply about design. The bootcamp also has the salubrious effect of setting students expectations, and calibrating instructor's notions of where each student's hurdles are likely to arise. Sometimes, the instructors learn that a project has to be toned down, or is too easy because a particular library already has all the important functionality.

Development. Students typically spend between mid-October to mid-March in intense development. This phase also contains the student's novel contribution, especially the algorithmic component described earlier. Students often discover that they underestimated some aspects of their project, and have to recalibrate or redesign. We believe this is a valuable lesson for the students to learn, akin to the unanticipated hurdles faced in any real-world project. We also find that many students experience a turning point before which they struggle with the different components, and getting packages to work. Once the pieces of their project start to coalesce, they get more productive and progress is easier to see. During this phase, some students also learn some significant new material in computer science. For example, students with a project involving computer vision have almost no prior coursework in this area. They must then learn how to use computer vision packages and yet make an algorithmic contribution on top of what is provided by the library. Many students rise to the occasion and surprise the instructors

with their results. A key component of any senior design course is that students enhance their ability to learn and apply computer science concepts independently.

Road show. Finally, the "road show" phase focuses on presenting the project in a number of mediums and venues. This includes posters presented at an end-of-year CS showcase and for parents leading up to graduation; and a final, eight minute presentation of their project to a broad audience (we invite other faculty, alumni, and all CS students) with an emphasis on giving a high-level motivation and description of what was accomplished. Throughout the year, we schedule practice presentations for the students all with the aim of culminating in a strong final presentation on "D-Day," the day we invite the whole department to watch senior design presentations. The best presentations are then invited to compete in the school-wide competition for the best senior design award. We assign two senior design awards within the department as well - one for best project and a specially-endowed award for the best entrepreneurial project.

3.3 Design and software engineering

What is probably familiar to most senior design instructors is that design is hard to teach. Our approach is to combine three processes with the hope that students develop a sense of design. The first is to gain an appreciation for the need for design. Merely telling them it's important has been shown, through prior experience in the department, not to work. Thus, when they start working through the bootcamp and assembling various components, the complexity of their project dawns on them, an important part of the appreciation. Soon, towards the end of the Fall semester, their code starts to become unmanageable without proper decomposition.

The second process involves the structured approach to design. This has three steps. Students are asked to carefully design and attend to all aspects of user interactions. After that, students are required to provide a functional design, listing the major components and their functionality, and their interactions. Typically, these components include the front end, a backend, database design, and the interactions with libraries. Finally, in the third step of design, students provide a detailed algorithm or pseudocode-level design, attending to key data structures. After this last step, students also need to design the organization of their code, and are encouraged to plan for modular testing (such as unit tests). This way, as students grapple with the complexity of their project, students are forced into design elements in a staged manner starting from the easiest to understand (user interactions).

3.4 Writing

Prior to 2010, students in senior design wrote design documents, consisting mostly of bulleted lists and diagrams. While useful, these reports were not graded for narrative style and were aimed solely at a technical audience (the instructors). At the same time, the university's writing requirements changed to include two writing-in-the-discipline requirements, one of which was

associated with senior design for CS students. These university requirements specify that at least two writing assignments include a process of revision. This shift towards a mix of technical and persuasive writing reminds students that after graduation they will not only be writing code; they will be documenting and presenting their work to a wide range of audiences.

3.5 Presentations

Over the course of the year, each student makes around 10 presentations culminating in the final one. Their training for these presentations starts during the bootcamp phase with an intense, full-day workshop on presentations. The intention is to give students some basic training in presentation skills, including strong eye contact, the effective use of gestures, modulation and projection of voice, posture, and to help them recognize their use of fillers such as "um" and "like." This sets the stage for the evaluation of all successive presentations. The students presentations are recorded, and after they present and hear our advice, they watch their performance to better understand the recommendations.

Throughout the year, the students give short, frequent presentations, each time with a subtly different focus that is appropriate for the status of their projects. For example, initial presentations include those that make the "pitch" for the motivation and contribution of the project as if to venture capitalists, or deans. Next their presentations describe the high-level design as if as a program manager to a team of developers. Toward the end, the focus is on honing the story and evaluation of their project for presentation to a general audience. Presentations are typically eight minutes long, forcing students to be concise and well prepared.

Providing feedback to help students improve presentations was initially a challenge: immediately after their talk, students are usually still somewhat stressed, and have trouble recording and remembering verbally given suggestions. Similarly, written comments are often forgotten or ignored. Our recent solution is for instructors to make notes in a shared document as students present, and require students to provide a written response with a concrete plan to improve their talk within a few days. Instructors reference previous feedback when watching a student to ensure that they don't repeat past difficulties, and that they are continuously improving.

4.0 Lessons Learned

In this section, we focus on lessons learned that might be useful for other institutions. These are based on our own experience as instructors as well as feedback based on exit surveys, but more importantly, from alumni who experienced the new interventions listed earlier.

What works, according to alumni:

- Almost to a person, alumni are enthusiastically supportive of the emphasis we place on presentation skills. They constantly find themselves having to make presentations and are grateful for the experience.
- A subset of alumni have also asked if we would increase the amount of required writing, explaining the importance of strong writing skills as they rise through management.
- Almost all alumni in post-graduation surveys rate the senior design course as one of their most valuable experiences in the curriculum.

What works, according to instructors:

- Doing short, frequent presentations with a clear feedback loop allows students to iterate and improve their presentation skills much more effectively than doing one or two longer presentations over the course of the year.
- The course's structure – significant faculty involvement during the bootcamp phase leading to a less structured relationship as students progress – helps students learn the independence they will need in the workforce. Students become visibly more confident over the year in both their technical and presenting skills.
- Even when students work on independent projects, each year's cohort builds strong bonds through the shared experience of working on a challenging large-scale project. Having a dedicated departmental space where seniors can work, debug, and practice presentations together helps build community, or as cheerfully reported by a recent alum: "Misery loves company!"
- Students build a strong relationship with their faculty mentors, and the course provides an avenue for local alumni to be actively engaged with the department.

What doesn't work? What challenges remain?

- A junior year pre-senior design semester did not help: it caused premature initial enthusiasm that disappeared in the summer and did not re-emerge in the fall. Students complained that they rarely pursued the projects they began in the design semester, and they could not see its benefits.
- Balancing the desire to give students independent ownership of their own project versus the benefits of working in a team remains a challenge. As class sizes grow, we anticipate a greater shift towards teams, but remain concerned about the loss of personal motivation and potential workload imbalances found in student groups.
- The course is resource intensive in terms of faculty-student contact time (3.5 hours per week for the lead instructor, 1-3.5 hours per week for faculty mentors, 2.5 hours per week for the alumni adjunct); while we believe this level of engagement has been important to the success of the program, it does not scale well when class sizes grow; the largest number of students a faculty member can be expected to mentor is about eight.

5.0 Conclusions

The capstone senior design course at George Washington University has evolved over fifty years to provide a challenging software development experience that will prepare students for their future careers. Our relatively small class sizes ranging from twenty to forty students give us the opportunity to run a course where faculty are deeply engaged with students working on independent projects. In addition to requiring projects with a deep algorithmic component, the course has a strong focus on verbal and written presentation skills. While many students initially scoff at such soft skills, our alumni report the course as being one of the most valuable and memorable experiences of their undergraduate careers precisely for that reason.

Acknowledgements
We would like to thank Emeritus Professor Arnold Meltzer for his long and inspirational commitment to senior design from its earliest days in the 1960s all the way until his retirement in 2004. We also thank Prof. Robert Lindeman (formerly at GW, now at Worcestor Polytechnic) who with Prof. Meltzer, helmed senior design from 1999-2004. We are also indebted to the enthusiastic participation of several alumni and industry partners: Kunal Johar (a former alumni instructor), Krista Bacungan, Peter Chen, Olga Chen (née Gelbert), Zohra Hemani, Jeff Moore, Amir Raminfar, Herve Roussel, and Zack Schwartz.

References

[1] J.M.Fursman. Designing a Capstone Course: A Literature Review to Support the Capstone Course in Defense & Strategic Studies. *USMA*, West Point, 2012.

[2] *The Green Report: Engineering Education for a Changing World*. ASEE, 2010.

[3] R.C.Hauhart and J.E.Grahe. *Designing and Teaching Undergraduate Capstone Courses*. Jossey-Bass, 2015.

[4] H.Hoffman. *The Engineering Capstone Course: Fundamentals for Students and Instructors*. Springer, 2014.