# CSci 487 Senior Project, Spring 2018 Postmortem

H. Conrad Cunningham

22 May 2018

## Senior Project Postmortem

The following aggregates the items I listed in the series of emails with the subject line "Senior Project Postmortem, Part n". I preserve the stream-of-consciousness order of items from those emails.

0. (Added) Changes made in Spring 2018 from most previous offerings
   - assigned a TA to help the instructor with his 32 students
   - dropped the separate Bibliography assignment
   - replaced the License Agreement assignment with a modified version of ORSP's Undergraduate Capstone Survey form
   - divided Design Specification into an initial and a revised submission
   - required explicit specification of a Minimum Viable Product
   - required use of version control software
1. Research-related projects

   If a student undertakes a research-related project rather than an application (as more and more are doing), then the student's final report probably should include some appropriate record of the "research product" not just materials about the software/hardware system assembled to carry out the research.

   For example, if the student carried out an experiment and collected data, there should be appropriate plots, analyses, etc., in a short "scholarly" paper as a part of the final materials.

   A special case is probably an Honor's College student whose thesis and project are combined. We probably want the thesis as well as the usual documents requested.

   We may want to talk about this issue in a faculty meeting early in the Fall.

2. Version control

   One new **required** component Armando suggested this semester and I made part of the course was use of version control (e.g., git using a server on the network).

   The result was mixed. A number of students (who had 387 in 2017 or 2018 or who just learned it for 487) did well, a few used version control some but not all that effectively (infrequent commits or many just at the end) and a few chose not to use version control. This latter group including some who had 387 but maybe did not learn effective use of version control, some who did not have 387 (e.g. BA majors) or took 387 in 2016 or earlier, and some who just did not want to be bothered with this requirement.

   In general, I think requiring use of version control was a reasonable step (e.g., to avoid disasters when laptops die and perhaps most of the students will need to use it after graduation).

   But many of us (including me!) need to learn how to use it or use it more effectively. We may need to introduce some explicit version control training.

3. Turing software (for Web and database development)

   A few students use our server Turing to develop web applications that may be migrated to client servers. At least one student commented that the older version of PHP on Turing caused some issues because his client had a newer version. (I mentioned this to Jeff before I sent the orginal email.)

   We probably need to better define what software we plan to support on Turing and have that software base on the same kind of regular update schedule as the Adler labs.

4. Requirements elicitation and analysis

   Some of our students are reasonably skilled at eliciting and defining the requirements for fuzzily-defined software and database projects "in the wild". Most are not.

   Some worked for half the semester or longer on the wrong things.

   We do not have a systems/requirements analysis course. I suppose we cover some aspects of this in 387, but not all students practice that in group projects where others probably take the lead on particular aspects.

   Perhaps as we define the new core course and shift some content around, we need to consider whether we can address the requirements elicitation and requirements/domain analysis issue.

5. Remote sponsors

   5 of the 32 projects had sponsors outsideof Lafayette county – 3 where students came of up projects on their own (McConnell, Logan, Rankins)

and 2 from the list that we suggested (Conway, Calhoun). All of these had some issues related to the remote nature of the project, although these issues could also happen on campus.

We may want to consider to what extent we want to allow or encourge such projects in the future.

6. Sponsor/user interaction

   In most cases, there was limited interaction between the sponsor and the student. Most sponsors complained about this on the evaluation. Much of this is that the student did not choose to meet with the sponsor much – because they were making slow progress or thought the project was easier/different than it was or the student was really the sponsor himself/herself. In some cases sponsors were hard to get a response from.

   Perhaps the instructor should talk with the sponsor up front if not a person we know.

   Perhaps there should be more required interactions between student and sponsor.

   By the way, students who got feedback from a variety of users tended to make the product better (e.g. Jesse Coleman).

7. Deployment systems

   In many cases (particularly for web applications) students and sponsors did not give sufficient attention to where the system would be deployed – despite my explicit warning about this problem.

   At the end of the semester, we still had a number of projects on either Turing or local-hosts with no deployment target yet identified. Some of these may never be deployed as a result.

   Sometimes this resulted in poor technological choices on the part of the student. Students chose a language or framework that was not supported on the likely available servers.

8. Testing

   In general, students did not test the systems sufficiently well.

   This was particularly a problem where the project was a piece of a larger system that was under development by others (e.g. McConnell) or the project involved enhancing an existing system (e.g. Feaster).

   I found in CSci 450 last semester that many of our students are not quite sure how to go about testing systematically (even though many had used JUnit some in 211 or other courses). The ones who have had internships in software development shops do this better than those who have not. (In CSci 450, I expected the students to test their Haskell program. I did not provide a testing framework or give explicit instruction on its use.)

9. Selecting a project early

   Most students wished they had selected a project before the semester began.

   Of course, that would mean that the project selection is less under the control of the course instructor, unless the instructor wants to give involved earlier.

10. Smaller, quicker, more specific deliverables

    My version of the course had the usual deadlines for certain documents and gave general guidelines for those documents, but some students wished for more specific guidance to keep them on track.

    We can probably do some better in defining what we want, but dealing with their own schedule is part of what we want students to master. And, of course, each project is unique, so the specific deliverables differ among them.

11. One-on-one meetings

    I had Armando as the TA. We scheduled two required meetings with Armando. He and I also have open office hours. I held the class meeting and asked for verbal reports 2 or 3 times.

    Some students failed to show up for at least one of the meetings with Armando. Others commented that a 3rd meeting would have been helpful.

12. Journal entries

    I used the Blackboard journal feature but gave few guidelines to what to report. (And I did not examine the journal entries sufficiently frequently.)

    Perhaps replacing this free-form journal with a list of specific questions they are expected answer would be helpful.

13. List of resources for student projects

    At least one student suggested an up-to-date list of Department or web resources (tutorials, tools, etc.) available to support student projects would be helpful.

14. Choosing technologies

    Some students are good at choosing technologies to use. Many have to meander around quite a bit before they settle on the technologies to use – often changing after the design spec.

    Some students try to learn too many new technologies and then use them to carry out the projects. Some did this successfully, but others used so much time on trying to learn the tools that they had little time to actually do the project.

A few might have been overly conservative, choosing only technologies that they were familiar with. They could have had more interesting projects and could have learned more if they had taken a few more risks.

15. Unclear schedule

    A few students complained that I had not made the schedule sufficiently clear.

    Part of this stemmed from the changes we made to the course and my lack of making all the needed changes at the beginning of the semester. ("My bad".)

    The schedule was on my website and I made Blackboard/email announcements, but I hesitated bombarding the students with too many routine announcements.

16. Proactivity

    Some students were insufficiently proactive when dealing with non-technical sponsors. They tried to deliver what the sponsor wanted, without necessarily exploring design/technology options and presenting them to the sponsor. (Of course, some students chose technologies without considering issues such as deployability or long-term support.)

17. Bibliography

    We eliminated the Bibliography assignment, which I think was good in theory, at least in that it encouraged students to push into design earlier.

    However, most students did not document their Final Report in a scholarly fashion with a list of references (despite an indication they should do so).

18. Documentation of code

    Most students do not document code internally.

    Perhaps we do not stress this as much in our courses as we did a quarter century ago.

19. Database

    Most students do not do a particularly good job of analysis and database design. At least more students seem to have had database or were in it this semester than in some past years.

    Armando's suggestion of the use of the Firebase NoSQL database seems to have been good for those doing mobile apps and some web apps.

    I hope the new core course will help this aspect in the future.

20. Design Specification

    With the elimination of the bibliography assignment, I split the design spec assignment into an Initial and a Revised Design Specification. I

think this was helpful. (Armando gave feedback on the initial spec; he, the other students, and I gave feedback during the mid-semester Design Presentations.)

At Armando's suggestion, we also required a *Minimum Viable Product* MVP) in this spec. Good idea! But many students had trouble knowing what this meant and in defining one for their project – defining either too large or too small an MVP.

21. Undergraduate Capstone Project Form

    I required this after Spring Break. It would have been better before Spring Break along with the Revised Design Specification.

    It was hard extracting these from some of the students. One student of the 32 never did it.

    I did not send these to ORSP. In maybe 4 cases where the sponsor kept the intellectual property, I should have.

22. Web design

    Some students spent waaaay too much time on the front-end, leaving insufficient time to do the server-side.

    Some students felt poorly prepared to do the full-development of a web app – client side, server-side, database, deployment on a server – from the courses they had taken.

23. Final submission

    We have required submission of the final report as a bound paper document with the source code on some electronic medium such as flash drive, CD, or DVD.

    I had this in writing, but I probably did not remind the students as much as needed. So there was some confusion.

    I like to grade from paper, but there are a lot of documents to manage when the class is 25+ as we have had most semesters for the past 3 years.

    Perhaps if we require version control in a place where the instructor and sponsor can access it, the submission of electronic media can be eliminated.

24. Seeking help

    Despite Armando and me being available during our office hours, few students came. ... However, many students bugged Kristi or Dawn about databased design. Thanks for helping them!

    Some students were almost pathological about trying to do everything themselves, without asking for any help. This resulted in some students risking complete failure, when a little bit of help could have made them quite successful.

Of course, we sometimes have the opposite problem where students are helped too much or students who copy the work of others.

25. Curriculum issue: Object orientation

    One student made an interesting comment: He felt he had been taught the "how" of OO programming reasonable well, but he felt there had not been much talk about the "why" of OO programming.

26. Curriculum issue: Systems courses

    A few students complained about the lack of systems courses – particularly "practical" systems courses where they would know more about system admin issues related to things like deploying web apps, etc.

    Maybe more frequent offerings of CSci 323 will help this.

27. Curriculum issue: Required internships

    (: One or two students suggested there should be "required" internships. :)

28. "Close to the bare metal" programming

    Several students did projects where they needed to understand and develop software that worked at a low level.

    Those that worked with Adam seemed to handle that aspect sufficiently well, but those that worked with outside sponsors struggled with that kind of programming. Saying, in retrospect, they did not feel they were well prepared by their coursework for their projects.

    One (Nguyen) handled that very well! Another (Conway) had mixed results.

    Can we prepare our students better for such projects?

29. Doing senior project with very heavy workloads

    One student was taking 23 hours, working part-time, and commuting from Tupelo. (Needless to say this student did to attend class frequently.)

    Another student had an ordinary load but was working essentially full-time. Others had heavy coursework including courses like 387, 433, and 475. Etc. Some of the latter were students who changed to CS from other fields late in their programs.

    I am not sure we can address this point directly – except advise students that they need to allow plenty of time to work on their project when they sign up for the course.

    Also see next item.

30. Curriculum: More frequent offering of core and popular elective courses

A number of students commented that they did not get the opportunity to take courses they would have liked to take because of infrequent offerings and full classes.