Department of Computer Science and Software Engineering

Senior Project 1
CSSE 497 Fall Term 2012

Instructor(s): Michael Wollowski
Email: wollowsk at rose-hulman.edu
Phone(s): 812-877-8650
Office: F218
Office Hours: Check with Senior Project Advisor at first meeting.

Introduction: Software projects are challenging efforts and as CSSE students prepare to enter the workforce, experience is needed. In the Junior Sequence of courses (CSSE 371/372/374/375) students gained experience with key aspects of software engineering as they completed a year-long junior project. Topics covered included:

1. Software requirements analysis & specification
2. Software project planning & management
3. Software architecture & design
4. Software construction & evolution
5. Forming and managing project teams
6. Software risk planning and management
7. Software configuration management
8. Software validation & verification

Senior projects follow a similar approach, but the students play a much more directive role in their projects by working with the clients to define the projects (typically 1400-1600 hours of effort for 4 team members). Rather than meeting in class, there are weekly meetings with the faculty advisors and the clients. The focus for the students is largely on the full development of a software system for a real-world client where the students define, plan, develop and deliver a usable product. CSSE 497 is the first of the three terms for this year-long project. Students are graded each term for their performance and participation on the teams, as well as using input from the clients.

Course Description: Group software engineering project requiring completion of a software system for an approved client. Tasks include project planning, risk analysis, use of standards, prototyping, configuration management, quality assurance, project reviews and reports, team management and organization, copyright, liability, and handling project failure.

Learning Outcomes: Upon successfully completing this series of senior project courses, a student should be able to.

1. Demonstrate effective Communication Skills
   - **Reading:** read technical documents and offer constructive criticism of their content and style
   - **Writing:** write several different types of technical documents
   - **Oral presentation:** prepare and deliver technical material at the appropriate level of detail
2. Demonstrate effective Management Skills
   - **Leadership:** lead a small software team (if team leader) or ability to support the leadership of the team (if not)
   - **Time management:** estimate and monitor personal time across multiple tasks
   - **Meeting facilitation:** lead and participate in small groups in constructive meetings

- **Estimating:** estimate effort required to complete technical tasks
- **Risk:** assess project risks and plan mitigation strategies
- **Planning:** prepare a feasible plan for the accomplishment of several technical tasks
- **Monitoring:** track the progress of several tasks according to a plan

3. Apply Technical Skills on a real-world problem.
   - **Analysis:** analyze technical requirements and proposals for feasibility and to model the consequences of proposed solutions
   - **Design:** construct appropriate abstractions of problems and solutions
   - **Coding:** produce and inspect implementations of software according to project standards
   - **Testing:** prepare test plans and to participate in both unit-level and system-level testing activities
4. Demonstrate key areas of Professionalism
   - **Ethics:** identify and prevent unethical professional behavior
   - **Intellectual property issues:** make appropriate professional judgments regarding choice of methods for protecting intellectual property
   - **Social issues:** evaluate and avoid possible negative social aspects of a software product
   - **Relationships with clients:** interact with clients in a professional manner

Prerequisites: CSSE 371. (*Software development and/or maintenance experience, and an ability write and communicate effectively will make this course more meaningful.)*
Textbooks: Readings may also be assigned from relevant papers (e.g., case studies).

Grading:
Your grade will be based on the amount of effort you put into your project and the quality of work you produce. Client feedback will also be solicited.

Expectations: Students will be expected to attend and participate in senior project related meetings. Announcements and assignments will be conveyed via Rose-Hulman email addresses and/or posted on the website. Students will be expected to work on most assignments with other team members.

Assignments: Project assignments are somewhat different than our usual classes. Please provide access to your software project material/artifacts in your configuration managed repository to the instructor. While this course is demanding, it is also rewarding for those that want strong understanding of software engineering as a discipline.

Academic Integrity: CSSE Honesty Policy (see http://www.cs.rose-hulman.edu/index.php/courses-mainmenu-28/82-honesty-policy.html) governs class and performance. Joint study is allowed (even encouraged) on some items as expressed by the instructor; however, each student must produce his or her solutions individually. Students must not collaborate on tests or homework that is passed in unless directed by the instructor.

Attendance Policy: Attendance of meetings is mandatory (unless with a legitimate excused absence such as illness). If you cannot make it to meeting, you are still responsible for all information covered in the meeting. Students who have more than 2 unexcused absences will receive a final course grade reduced by up to on full grade

level; a student whose total absences (excused and unexcused) exceed 8 will fail this course.

Valid Excuses: A valid excuse consists of a memorandum on Institutional letterhead from the Dean of Students. Job and graduate/professional school interviews, attending scientific conferences and Institute-sponsored activities are also valid excuses provided that every attempt has been made to avoid missing major assignments and examinations, and the student notifies the course instructor in writing at least one week in advance of the event. Illness and exceptional circumstances are, of course, valid excuses if a confirming memorandum from the Dean of Students is provided within one week of the illness/circumstance.

Writing: Written communication is important in CSSE 497, as it is in the software profession in general. Remember that a software document has several unique and important characteristics:

1. Technical documents are often the result of group authorship, thus it requires planning and final tweaking.
2. Specificity and organization are more important than flow; hence technical documents are often ordered around lists and tables rather than paragraphs.
3. Documents are often the reader's only source of information on the particular subject or product; hence they must be thorough and complete.
4. Documents are often used to answer specific questions; hence, they should facilitate finding specific pieces of information (navigation).
5. Documentation must bridge from general specifications to particulars of implementation and operation, hence it must make abstract concepts concrete and make concrete facts fit generalized concepts.
6. Documentation can be presented in many forms: online via HTML, MS help files, just plain text, and on paper as reference manuals, tutorial, quick reference guides, etc. It is important to choose the correct medium and even more important to write to fit the medium.

You can always drop by your project advisor's office if you have any questions regarding your documents. I would be happy to look at it and suggest improvements. You should also be aware of the service provided by the Learning Center.

Professionalism: As would be expected in the workplace, you are asked to behave in a professional manner. This includes your appearance, such as your apparel, your hygiene, as well as potentially offensive computer desktops. We furthermore ask that you turn off your cell phones during meetings, let people finish their turn talking and do not be disruptive in other ways. Violations of this policy will have a negative impact on your "Meetings" score.

Caveat: The instructor reserves the right to modify the course content, schedule, policies, etc. outlined in this syllabus.

Senior Project

Fall deliverables

The following are brief descriptions of the fall deliverables along with some relevant instructions and recommendations. These deliverables are a starting bid. If your team feels that any part does not apply to your project, please make your case.

1) Engineering Journal

Each team member has an electronic log, documenting what you did, how long it took, decisions your team made, and how those were made. The idea is to keep adding to this daily and weekly. The instructor will be periodically looking at these (typically weekly).

We will set up a team blog that the instructor will administer. We expect posts to happen when you do any non-trivial work on the project. The blog will not allow retrospective/prospective posting (sending posts into the past or future).

2) Configuration Management Plan
   - Similar to CSSE 376
   - Probably use GIT / GITHUB unless some requirement drives something else
   - Course instructor must have near admin rights
   - **System due end of 2nd week unless complicated**

3) Problem Statement
   - Should have a problem statement agreed to by your client by midterm.
   - Should be updated with changes at end of term.
   - Normal format is for the problem statement to be about 2-3 pages, to include a *short list* of key features, quality attributes, and other pertinent info the client agrees to, like how the system relates to or replaces some existing system. The Function Form Economy Time model used in 333 would be a good template.
   - **Preliminary Version: 3rd Week**
   - **Final Version: 5th Week**
   - **Living Document - This should be up to date, so you can use it to discuss priorities with your client.**

4) Project Plan
   - Description of your processes (e.g., pick a lifecycle, explain how you will use it, and why you expect it to be a good choice)
   - Schedule
     - Who's doing what (coding, testing, etc.)
   - Risks
     - Updated and added to (includes all risks)
     - ID
     - Type (risk/opportunity)
     - Probability
     - Impact
     - Mitigation strategy with revisit date (for all risks)
     - Outcome
     - Resources responsible
   - Overall completeness
     - Includes structure for subsequent quarters (we will provide our expectations ahead of time, toward this end)

    - Uses good documentation standards
- **Preliminary Version: 3rd Week**
- **Living Document - The "plans" should turn progressively into a record of what you have done, in sufficient detail that we can trace your activity (week by week) versus plan.**
- **Note: All elements of the Project Plan should be observable by the course instructor.**

5) Metrics
   - Identify Product, Communication and Process metrics
   - Track Product, Communication and Process Metrics
   - Maintain a Dashboard of Metrics on the project website
   - **Preliminary Version: 2nd Week**
   - **Living Document - Should reflect current risks and your plans to fix those! (And also show the history of this activity.)**
   - **Notes: less can be more in this arena. Have some justification for each metric selected.**

6) Requirements Document
   - Requirements will normally include detailed use cases and a supplementary specification, based on input from your client and from other sources. Occasionally, other pieces also are important. (E.g., if you talk to other systems, perhaps charts of those interactions.)
   - **Preliminary Version – 4th Week**
     - Should have a draft of requirements to discuss at the **4th week** presentation
     - Should have a draft turned in to a designated 371 team by end of **5th week**.
   - **Client Signoff – By 7th Week**
   - **Requirements Review – 7th Week**
   - **Living Document - These should be updated every time you negotiate a change with your client. After they sign off, it should be "a controlled document" with a record of when it was changed and why.**
   - **Note: if using an iterative process can keep a 2-3 month requirements/use case backlog, as long as it is replenished appropriately.**

7) Requirements Presentation

   - What the project is about?
   - The problem statement and requirements you have gathered and analyzed
   - The process you've decided to use, and why that's right for this project
   - Known project risks, and possible plans to deal with those
   - Feature Listing and an exemplar of functional specification technique (use case, scenarios, etc)
   - Current status
   - Anything else of interest to other teams.
   - **Week 4 – Wednesday 6th and 7th Hour**
   - **10 -15 Minutes per team – Do not spend excessive time preparing for this. The content should come straight out of your documents.**

See http://www.rose-hulman.edu/class/csse/csse497-498-499/Process/ for more details on the presentation

8) Initial Architecture Document

- This is a trial at a high-level architecture. It might be changed as you try to detail it bottom-up during development, as you investigate components, subsystems, and other low-level aspects of the design work.
- It should include at least:
  - An explanation of how why this design will solve the problem described in your problem statement
  - A high-level static view of the structures you expect to use, like the class diagrams you did in CSSE 374
  - A high-level dynamic view showing how the system will operate as software - like what major components communicate during the main operations
  - A high-level "allocation" view showing how the running system will sit on hardware and what these boxes will exchange to make the whole system work
- **A design document that you can explain, as a team, to your advisors and/or to your client (to the extent this is appropriate) - Due Week 10**
- **Living document - This document needs to change as you change your design. For most teams, this is likely to grow into being the "design document" as well. Like the requirements, once you have a "baselined" version of this, it should be under some kind of change control.**

9) Prototype

You should select some aspect of your system to prototype. This should be based on reducing risk, showing that you can do some difficult part of the technical work. For example:

- Prerequisite: a draft of the "Initial Architecture Document" – the prototype should be appropriately within this "initial" design.
- A usability prototype, which tests whether humans can operate the robot as desired. If you can't show this on real hardware, a fallback position is something like a paper prototype, like you studied in CSSE 371.
- A functionality or reliability test for some part of the design.
- **A working prototype, shown to the client – Week 10**
- **If you use Agile processes, this prototype is likely to grow into the real product, and be shown periodically to the client as it grows; this would mean that before long it should reflect the real architecture of that product.**

Weekly Meetings:

Each week, the project team will meet with both your client and project advisor. By the end of the first week, you will provide the day and times for these meetings to your advisor.

For each weekly meeting with the team's advisor, a message must be sent in advance of the meeting (i.e., by 9:00am) indicating the following:

1. A list of tasks completed this week along with how much time they took
2. A list of tasks planned for next week along with an estimate of how much time they will take,
3. A list of Red and Amber risks

4. A list of issues/requests for resources/observations relevant to the project.

A bit more on your overall strategy: After the team has finalized a version of the requirements, the project team will dictate the strategy they would like to pursue (i.e., what life cycle approach). Please determine this based on the project and client expectations/constraints. Typically, teams have switched to a more agile/iterative technique and designed, constructed, and tested a prototype that demonstrated their ability to advance the project. A preliminary architecture/design/test document will be very useful if the team decides to venture down this path. While this in not yet mandatory (see CSSE 498), it will also help the team to finalize and document the architecture/design/acceptance test plan for the complete system by the end of the first quarter.

We recommend that the team develops a "Vision document," which is internal for the team's use, which describes conceptually what you'll do over the whole year and what this will accomplish for your team. It should have the start of a "life cycle plan" in place by end of term, describing what you'll deliver at end of winter term, and spring term, and how the client will manage after that. This should include:

- Plan for getting and using customer feedback
- Plan for supporting the customer