Project Companion
Software Engineering Concepts (ECE473-573)

Jonathan Sprinkle

Spring, 2011

http://www.ece.arizona.edu/~ece473/

Document Description:

This Project Companion will help you understand the specific requirements, deadlines, and expectations for your project for the semester. The project's grade weight is specified in the syllabus.

Graduate and Undergraduate Projects:

Graduate and undergraduate projects will differ in complexity, difficulty, and expectations. Graduate projects are expected to integrate novel ideas, or be demonstrations of proofs of concept that are extensions to the state of the art. Undergraduate projects may be alternate implementations of known, engineering-related, problems.

Undergraduate teams are limited to three students. Teams may solicit help from the instructor in determining which algorithms to use, where to find best-practice implementation guidelines, etc. Graduate teams should find this information on their own.

Graduate teams are made up of a single student, and are expected to solve more technical problems on their own. All graduate projects should provide a sound backing in their documentation for their methods and implementation, by citing relevant research papers, books, or experiments. This standard will be higher for graduate projects than for undergraduate.

Project Overview:

You must produce an application for either iOS*, or Android. The selection of this platform is the same for all team members. Teams are encouraged to select a platform for which at least one team member has personal access to a physical device, as physical devices available to the team for checkout are limited.

Your team will design, implement, document, test, and validate your application against a set of requirements *that your team decides and the instructor approves*. The effort of this project will be distributed throughout the term.

Specific, example project topics should be explored with the instructor prior to the due date for project topics. This is to provide teams with the experience of "carving out" a project for the platforms. Project topics should integrate the multiple features of the platform, rather than simply develop an information-based client. Platform features include the accelerometer, camera, GPS, multi-touch (multi-finger) feedback, etc. Integration of one or more of these features for a project is required, unless sufficient novelty in the application makes this requirement unnecessary (as approved by the instructor). An abbreviated list of topics, and their chances of being accepted by the instructor, is provided on the course website.

*iPad, iPhone, iPodTouch, etc.

Timeline

For each project assignment due through `d2l`, only one project member is required to submit. It is encouraged that submission be performed in advance or as a group, so that group members can validate submission. There is no penalty for submitting early, and replacing submission. For all releases, the source code will be extracted from the revision control system *by date*, unless a specific revision number is provided by the team.

- F 1/21: Project teams due to `d2l` by 5:00 pm
- W 1/26: Project topics due to `d2l` by 5:00 pm
- F 2/4: Project requirements due in PDF form to `d2l` by 5:00 pm
- W 2/16: Project Design Document due in PDF form to `d2l` by 5:00 pm
- F 2/25: Updated Project Design Document due in PDF form to `d2l` by 5:00 pm
- F 3/11: Alpha Release due to `d2l` by 5:00 pm
- F 4/1: Team must have integrated objective measures to validate their list of requirements (i.e., requirements validation)
- F 4/16: Beta Release due to `d2l` by 5:00 pm
- W 4/13-W 5/4: Project Presentations (in class)
- W 5/4: Final release due in `.tar.gz` form via `d2l` by 5:00 pm

Project Requirements (and changes)

Each team is expected to provide a substantive list of *specific* requirements of their project. These requirements will be refined over the term by the team, **and are subject to change without consultation at any time by the instructor**.

In the "real world," changes to project requirements during the course of development are not only common—they are expected. In this course, the instructor will simulate the likelihood of external forces changing the requirements of a software project by enforcing changes in requirements or assumptions during the course of the project.

This can happen at any time during the semester, and may happen more than once for each team. The importance of regression tests (that analyze and confirm/question the implementation's ability to meet requirements) is evident here. **Teams whose design and implementation strategies are not robust to requirements changes will therefore expend more effort in adapting to requirements changes than teams who think carefully about the impact of requirements on their design and implementation.**

Project requirements are provided in 2 classes: requirements to qualify for a B, and to qualify for an A. The entire set of B-requirements must be met for a maximum grade of "90" on the project's implementation grade. A sufficient subset (determined by the instructor and team) of the A-requirements must be met for a maximum grade of "100" on the project implementation. Again, the team is responsible for coming up with these requirements, and demonstrating that they are met. Teams are encouraged to align their B-requirements with their Beta release, and A-requirements with their final release. **If tests are insufficient in demonstrating the satisfaction of a requirement, then the requirement may not be considered as met, and reductions may be applied.**

Design Document

The design document should describe the starting, and ending, points of the project, along with the requirements the project will meet, the expected time-based progress of the group for each portion of the project, and other relevant information. It will give the names of the project members, each member's responsibility (e.g., interfaces, classes, tests, etc., which will be written by each person) and the timeline for developing

the code, documentation, tests, etc. The design document (and its revisions) should be kept in the team's revision control system repository.

The design document should satisfy all of the concerns of an appropriate software engineering design. Consult with the relevant chapters in the SWEBOK, along with a rubric provided by the instructor, when preparing your design document.

Each team's project will be distinct; no two teams are expected to perform the same tasks, though two teams may be permitted to attempt to satisfy similar constraints.

After instructor feedback is provided, teams will resubmit their Design Document, having made the appropriate changes. Both the initial, and updated, Design Documents are expected to be the full document. That is, the first document is not a "rough draft". In other words, if the instructor has no criticism of your initial draft, it will be your final draft without any modifications, reformatting, filling in of paragraphs, etc.

Project Design Feedback

The Design Document represents the grading criteria each team establishes for itself. Thus, the instructors will give feedback to the teams based on the Design Document. Each team is expected to implement the modifications (if any) suggested by the instructor, and resubmit the Design Document.

Releases (Alpha, Beta, Final)

Releases will be provided to the instructor at specified times. Each release will be graded on its match with the schedule provided in the design document, the completion of its documentation, its compilation, and any tests which run as part of the release. In addition, the relevant documents provided by the team to explain how to run the project will be followed *exactly*. Failure to provide necessary documentation for the installation and execution of your project may result in a catastrophic grade.

All submissions to `d2l` must be archived by making a distribution, and submitting a single file that encapsulates that distribution to `d2l`. When grading project releases, the instructor will extract all necessary sources from the revision control system using the due date and time, or a specific revision number (if specified by the team). The archive on `d2l` will be used only if issues emerge from the revision control system version.

Requirements validation and regression tests

The team is responsible for establishing and validating that requirements are met. If a requirement is present in the design document, and the team does not provide instructions for how to validate that requirement as part of their release, then it is not considered met and this will reflect poorly on the grade for that release.

Teams may choose to have regression tests (tests that verify that bugs have been fixed, or requirements are met) for this portion of the validation, but the technology used will change based on the platform selected.

Project Presentation

Each team will demonstrate and present their project to the entire class. This project presentation should include the engineering and science in the project (not just be a verbal description of the code), and should stand on its own to future students on what it was that your team did. Running demonstrations must be part of your presentation, with the understanding that teams who demonstrate prior to the last day will certainly be demonstrating an "unpolished" work. Major deficiencies the implementation will receive poor marks.

Grading breakdown:

All team members will receive *the same grade*, except in the case of flagrant delinquency of a particular project member as judged by the instructor. Thus, it is *imperative* that team members specify to one another their schedules, preferences, strengths, weaknesses, and any other relevant information. Projects which miss a deadline will have that deadline grade deducted *for each team member* by multiplication of 0.63 for each 24 hour period that deadline is late—without exception.

- 5% — Project Requirements
- 10% — Project Design Document (Initial)
- 5% — Project Design Document (Updated)
- 15% — Alpha Release
- 10% — Requirements Validation
- 20% — Beta Release
- 10% — Project Presentation
- 25% — Final Release

Team Member Misconduct

In the event of flagrant misconduct of a team member, an individual's contribution to the team's effort may also be determined by several factors, including:

- satisfaction of requirements assigned to that team member;
- frequency and number of commits to the revision control system; and
- identification of source code files that were authored by that individual.

The need to calculate an individual's contribution to the team will be the exception, not the rule, when assigning project grades. Each project document (including all source code) should contain *only* the names of each student who served as an author for that document.

Academic Integrity

All project tasks are team tasks, and collaboration within team members is expected. Sharing of solutions or partial solutions to homework assignments between members of the same team is a violation of academic integrity, and will not be tolerated. Likewise, sharing of solutions or partial solutions to projects between teams is a violation of academic integrity *without the express consent of the instructor*. Consent may be given in the event that students from two teams are collaborating through a common framework, or a similar partnership established in advance.

Students may, at any time, share code through approved `d2l` mechanisms to be used by *the entire class*, as long as that code does not satisfy homework requirements prior to the homework being returned to the class. For example, the instructor may request that a student to post their solution to a homework assignment, after all homeworks have been graded, as a matter of demonstrating good code practices.

Issues of Intellectual Property

Please familiarize yourself with the Family Educational Rights and Privacy Act (FERPA). According to FERPA guidelines, the project work performed as part of this course may be taken without question by individual group members and reused after graduation. This makes it critical that you correctly acknowledge the authorship of each document in your project to its appropriate creator.

Any group of students may license their code using an open-source license. However, this license may not be implemented during the semester due to FERPA regulations, privacy issues, and academic integrity issues. Details of open-source licenses are covered as part of the core academic curriculum.

Thanks to FERPA regulations, the Arizona Board of Regents, the University of Arizona, and the instructor have no intellectual ownership of the material you will develop as part of this course. Thus, you as an individual may take the material for which you were author, or as a group you may take your entire project, and reuse it for the purposes you see fit after the conclusion of the term, with no requirement for licensing

or notification to ABOR or UA. Note that this right extends *only* to work done as part of your education; if you are paid for your efforts are part of a lab assistantship, FERPA does not apply.

Some projects presented by the instructor *may only be attempted if all students performing on that project waive their rights to exclusive ownership of the project material following this course.* The instructor and collaborators have previously established the intellectual ownership of these project topics. It is important to note that students performing on these projects *would* be able to retain their own rights to this codebase, but the instructor (and any collaborators in the future) would have similar rights. Details of these rights will emerge as part of class discussion, or can be explored during office hours.