**Software Engineering Term Project Guidelines**

Summer 2016  
Updated: 5/16/2016

**1. Introduction**

Real-world software development is different from the experience gained in most class programming projects. Requirements are often unclear and complex. No one can accomplish a fairly complex project singlehandedly. Systematic testing is vital. In the real world, one must work with the realization that the software produced will be used by, and maintained by, other people, hence usability and maintainability are important qualities. The purpose of the software engineering term project is to help you get a feel for real-world software development. Students will work together in teams of 4-6 members. Students will be taking this project from problem definition to system test, applying basic software engineering processes.

*You should understand that the project will require a significant commitment of your time from the beginning of the semester to the end.*

**1.1 Overall Objectives**

* Learn teamwork.
  + Task distribution, communication and coordination.
  + Learn to break down tasks and define responsibilities.
  + Learn to use supporting tools for team collaboration.
* Apply problem definition techniques.
  + Learn to elicit and specify requirements.
  + Learn to use object-oriented analysis techniques.
* Create solution systematically.
  + Learn to create design models.
  + Develop implementation that is consistent with design.
  + Employ a version control system.
* Gain experience with validation techniques.
  + Learn to conduct formal reviews and inspections.
  + Learn to plan tests early in the process.
  + Learn to use simple automated testing tools.

**2. Project Proposals**

* Develop a web application with community impact.
* Web application must make use of modern platform and toolset. Primary resource: [http://www.saasbook.info/home](http://www.saasbook.info/)

The project proposal is to be written up as a problem statement. This statement will be reviewed and the proposal may be modified in order to fit within the time constraints.

**3. Working Together**

Students will be asked to self-organize into teams consisting of 4-6 members.

**3.1 Grading Policy**

Each member of the team is expected to contribute equally to the project. **Note that project grades will be given on an individual basis.** In other words, if one person does not contribute substantially to the project, his/her grade will be significantly reduced. Weekly logs (see Section 4.2), peer opinions, project tracking data, and code commit data will be the primary sources of information for assessing individual contributions to the project. If there is a clear discrepancy in individual contributions, individual project grades will be adjusted accordingly.

**3.2 Team Roles**

Each team must appoint:

1. team leader (head honcho) - coordinate the distribution and assignment of work and serve as the point of contact.
2. scribe (record keeper) - record meeting notes.
3. architect (idea person) - main designer/technical lead.
4. quality assurance (bug catcher) - main testing lead.
5. tech support (genius bar) - support and educate the team on the computing environment.
6. software developer (code ninja) - participates in requirements, design, implementation and testing of software.

One person can have multiple roles.

**3.3 Collaboration Tools**

Our experience has shown that some CSCI 4830 projects fail because the teams were unable to make progress due to teamwork issues (unable to find common meeting time, unable to agree on ways of communicating decisions and tasks, etc.). To mitigate this, you are required to make use of tools for tracking project activities. Some commonly used tools:

* Integrated with version control:
  + Github
  + Google Code
  + Assembla
  + Trac
  + Rational Team Concert
  + Team Foundation Server
* Tracking only
  + Kanban board, e.g., http://kanbanflow.com
  + Trello
  + Pivotal Tracker

**3.4 Tips**

While there is no magic formula for successful teamwork, the following are helpful tips from my observations of successful teams in the past.

* Set up a regular team meeting. Take notes during the team meeting and post them. At the end of the meeting, make sure that each member is clear about his or her work items.
* Assign work items to individuals. Work items that are assigned to two or more members usually end up being completed by one person, or worse, not done at all.
* Record work items in a project tracking system. In this way, it should be possible for me to get an idea of each person's to-do list at any given time.
* In the meeting notes, record who were in attendance and who were not. A few absences can be tolerated, especially if there is some excuse (preferrably given ahead of time).
* Nonresponsive teammates disrupt the smooth functioning of the team and can wreak havoc on the morale of the team. Refer nonresponsive teammates to me. These include but are not limited to, (a) those chronically absent (3 or more times) with no excuse, (b) those who repeatedly fail to respond to queries, (c) those who continually neglect their work items.
* It is helpful to sit together in class so you can use the times before and after class for quick discussions.

**4. Project Artifacts To Be Submitted**

These are the artifacts that should be submitted. Examples and templates will be provided later.

|  |  |
| --- | --- |
| **Artifact** | **Percentage (Tentative)** |
| Requirements - Requirements specification document | 20% |
| Design - Design document | 20% |
| Implementation - Source code - Version control repository | 35% |
| Verification & Validation - Test plan - Test scripts - System test summaries | 15% |
| Management - Project plan - (Weekly logs) - Work item assignments - Project Summary Document | 10% |

**4.1 Notes on Version Control Repository**

Source code for the project must be version-controlled, i.e., modifications should be saved using a version control repository such as Subversion or Git throughout code development. Version control support is also available through Rational Team Concert. This repository is part of the final project submission.

**4.2 Notes on Weekly Logs**

When your team proposal has been approved, each team member must record every week a log of his/her activities contributing towards the project. The tasks should be brief and specific. Examples:

* Wrote use cases for the requirement querying functionality.
* Working on object model: (list of objects)
* Updated issues list.
* Modified file.c and checked into Subversion.
* Met with team. See posted meeting notes.
* Could not make progress because of ...

Fill out the weekly log here: <http://cs2.ist.unomaha.edu/~hsiy/SElogs>.

**5. Major Deadlines:**

|  |  |  |
| --- | --- | --- |
| **Date** | **Deliverable** | **Nature of Feedback** |
| 5/23 | Team composition + initial project ideas | Verbal approval |
| 5/31 | Problem Statement/Project Proposal | Written comments |
| 6/6 | Initial Project Plan | Written comments |
| 6/27 | Proof-of-concept or working prototype | Written comments |
| 7/5 | Initial Requirements Specification Document | Written comments |
| 7/12 | Initial Design Document, Code Milestone 1, Test Plan | Written comments |
| 7/26 | Code Milestone 2, unit test scripts, test summaries | Written comments |
| 8/12 | Final Submission (Code Milestone 3 + all project documents) | Grade |

**5.1 Notes**

* These are basic deadlines for every team. Teams wll be expected to work out a plan of internal tasks that are consistent with these deadlines.
* An acceptable version of each project artifact must be submitted by the deadlines above. Late or incomplete submissions will get a 1-point per day deduction from the final grade.
* If you are unsure whether your submission would be acceptable or not, please come and talk to me, *before the deadline*.
* When I provide feedback on a submitted deliverable, it is expected that the feedback should be incorporated into a revised version of the deliverable and resubmitted for re-review. *Expect to make multiple revisions, especially for the requirements and design documents.*
* When resubmitting a deliverable, include a brief item-by-item explanation how each of my comments were addressed.
* The proof of concept prototype due in June will be graded as an exercise. The purpose of the prototype is to help you understand the capabilities and limitations of the implementation technology.