Project Plan

Software Design **Allen B. Downey**
Spring 2006

It is time to start thinking about class projects! Here is the schedule of deadlines:

- now: project shopping.
- March 31: project proposal.
- April 7: design proposal.
- April 14: design refinement.
- April 21: demos.
- May 4: final design report due.
- May 17–19: Expo

Project shopping

We will have some time in class for people to present project ideas, with the goal of getting suggestions from the class and possibly finding people with similar interests to work with. You can also post messages on the class mailing list, `software_design@lists.olin.edu`, either looking for people or ideas.

Project proposal

Your proposal should be a short document (1–2 pages) with answers to the following questions:

1. Who's in the group?
2. What's the project?
3. What's the minimum/maximum deliverable?
4. What's your first step?
5. What's the biggest problem you foresee or question you need to answer to get started?

You should turn in one copy of your proposal for each project group. As the project proceeds, you will be editing this document and turning in updates.

A note on style: The documents you turn in at various stages do not have to be complete, or polished, or formatted beautifully. Think of them as successive drafts of the final report. But please don't take that as a license to make them incoherent, or verbose, or silly. You should try to present your ideas clearly and concisely, with the goal of producing a final report that you would be proud to show to the outside world.

Design proposal

At this point in the project you should have at least a rough idea of the design you are planning to implement, you should have chosen the packages you will use, and be familiar with them, and you should have written some code that at least tests out the packages. You might also have a prototype of some kind.

The design proposal should explain at least one part of your design by telling a story. The structure of the story should be something like this:

1. Here is a problem that we want to address.
2. Here are two or three ways we considered addressing it.
3. Here is the one we chose (at least provisionally) and why.

You don't have to document every design decision you made, but you should choose at least one that you think is interesting.

Also, your design proposal must include both a UML Class diagram and a UML Object diagram. If there are no classes or objects in your design, that might be a sign that you need to rethink your design, or your project!

Some projects may not involve any significant data structures or algorithms, but if yours does, you should describe them at an appropriate level of abstraction.

Finally, your proposal should include a development plan. What have you done so far, and how do you plan to proceed? How will the work be divided among the members of the team? If you are planning to split up the work, how will you manage the interfaces between the parts of the project? If you are planning to specialize, I suggest that you make some effort to ensure that all members of the team understand all parts of the design at an appropriate level of abstraction.

Design refinement

At this point you should have at least a prototype, and possibly you will have achieved your minimum deliverable. You should turn in an updated version of your design proposal that includes any improvements I might have suggested during the previous iteration, and any refinements you have made in your design.

In particular, I would like to know if there are any significant changes you made in the design while you were in the process of implementing it, and why. Examples might include:

- You added new classes to the design, or consolidated classes, or changed the relationships among the classes.
- You discovered a performance problem and changed your data structures and/or algorithms.
- You started out with the intention of using some package, but you needed something it didn't have or you found something better.

Also, this report should include a discussion of abstraction and language. As you develop your design, you should find yourself developing abstractions (like new classes and collections of functions and methods) and vocabulary for talking about your program, and for expressing the computation that makes up your program. What are the elements of this language, and how are they reflected in your design?

Final report

Your final report should include the following elements:

- All the elements from the previous documents, edited to bring them up to date. Sections that are no longer relevant can be removed.
- A final refinement of the design, including any changes since the previous iteration and an explanation of why the change occurred.
- An analysis of the outcome. Did you achieve your minimum deliverable? If not, why not? Did you achieve your maximum deliverable? If so, what went right (that is, why was the project easier to complete than you thought)?
- A reflection on your design. What was the best decision that you made? What would you do differently next time?
- A reflection on the division of labor. Were you able to divide the project in a way that allowed the members of the team to work independently and then integrate their work into a whole?

Of course, the organization of these elements will depend on your project, and some may more more applicable than others.

Demos

During the last week or two of class, you will have the opportunity to demonstrate your project for the class.

We will figure out the schedule later, but you should aim to be ready to demonstrate your minimum deliverable on April 21.

Expo

Depending on what else you are doing this semester, you might choose to present your Software Design project at Expo. If so, then each member of the team will make an individual presentation. It is possible that only some members of the team will present this project.

If you choose to present your project as a poster, it should include a description of the project as a whole, and also an explanation of what part of the project you worked on. This part of the poster should be targeted to a general audience with no programming experience. Your poster should also include an explanation of at least one design decision; this part of the poster can be targeted to an audience of programmers.

In my opinion, the most important criterion of a good poster is that it should pass the 15-second test. Someone should be able to walk up to your poster, know immediately where to start reading, and, in the first 15 seconds, they should learn what kind of project it is (that is, a software design project), who worked on it, and what you did.

After that (unfortunately) most people will walk away. For the ones who stay, your poster should tell a short, coherent story about your project. The flow of the poster should be obvious (that is, what to read first, and so on) and there should be a small number of conclusions (like one) that are clear and apparent. The total number of words should be small. The figures should be clearly labeled, and there should be text that explains what the figures are meant to show. You should design the poster as if you will not be there to explain it.