

Teaching Manual for CS2340, Objects and Design

Mark Guzdial

April 11, 2000

The purpose of this document is to be a guide to new teachers to CS2340 *Objects and Design*. It is *not* an introduction to object-oriented design, but a few pointers that may not be obvious from the catalog definition.

1 Learning Goals

Below is a bit of expansion on the brief learning goals associated with the catalog definition.

- A central focus of this class is on object-oriented design. We present a design process, discuss various designs, and teach specific methods and notations. In particular, we teach CRC Cards for OOA and UML class diagrams for OOD. We review most of the UML class diagrams, but students are only responsible for class diagrams (and not all features of those, just a core set).
- User interface software and human-computer interface design are issues in this class. The learning goals for UI software are for students to understand a traditional MVC-structured interface toolkit (and relate it to current UI software like Java's Swing); a more modern event-based and/or constraint-based UI toolkit (e.g., like Squeak's Morphic, but Thinglab would also do as well); and Web-based interfaces via CGI or servlets. The learning goals for HCI design are (a) simply to have students understand that designing interfaces for themselves does not mean that software will be usable by others and (b) to introduce them to issues of HCI design process and evaluation methods.
- CS2340 continues the exploration of language translation issues started in CS2330. Students are expected to learn in greater depth parsing and tokenizing, by practicing and exploring them in an object-oriented language.
- Again, along the lines of continued exploration of language translation, implementations of object-oriented languages is also a focus. We spend two lectures on optimization (e.g., exploring data structure implementations and how to time things) and two lectures on VM's (e.g., Squeak details, comparison to Java's VM).

- We use Smalltalk, specifically, Squeak (<http://www.squeak.org>) as a pure object-oriented language and not C-based, so learning Smalltalk is a goal. Squeak is chosen because it is free, cross-platform, and a focus of research here at Tech.
- Students should be able to use functional programming kinds of constructs available in Smalltalk, such as *select:* and *detect:*.
- Finally, as this is at the end of a series of programming languages introduced in the undergrad curriculum (i.e., Scheme, MATLAB, Java, C, and Smalltalk), students should be able to discuss tradeoffs between the various paradigms and specific languages.

2 Class Project

Most of CS2340 revolves around the single term, mostly-group class project. As the project changes from term-to-term, it means that the ordering of lectures is dependent on the project selection. The early lectures on introducing objects, Smalltalk, UI software, and object-oriented design will probably always stay at the beginning, but thereafter, choices can be made as befits the current project. For example, if students have to meet a deliverable that involves the Web before they have to meet a deliverable that involves parsing, the parsing lecture could be made later and the Web lecture earlier (which is the reverse of what happened during Spring 2000).

The desired characteristics for a CS2340 project are:

- The project should be complex enough that serious design skills and teamwork are required. A single student should not be able to complete the program in a single semester.
- Preferably the result should be authentic, actually useful to someone.
- The project should involve some parsing, to make the language translation aspects useful.
- Students should have to build more than one user interface for their application, to explore UI tradeoffs and to see the benefit of a model-view distinction in their system.
- There must be a *twist* at some point in the project, that is, a change or addition in the requirements that should be easy if the system is designed well but should require the students to re-think their design (and hopefully, see the value of a good design).
- Preferably, there should be some multimedia component to this, simply for the benefit of motivation and to take advantage of the multimedia facilities in Squeak (beyond what is normally in a Smalltalk).

As an example, the project in the Spring 2000 semester was to create a multi-column, personalized newspaper, built from Web-based news sources (selected by the

user), and generate Postscript for it. It's a complex project—in fact, optimal layout is NP-complete! It's arguably useful to someone. Parsing is required to pull out news articles from (ever-changing!) news sites. The twist during the Spring 2000 semester was to create a Web-based interface for selecting news sources and to deliver the newspaper on the Web as well as Postscript. Squeak's text support includes multi-column ("linked") text areas and Postscript generation, so it involved using some interesting multimedia support.

There are typically seven assignments turned in during CS2340, about two weeks apart.

1. First assignment is an *individual* assignment, so that *EVERYONE* has to do some Smalltalk programming.
2. Second assignment is building a user interface for their eventual program. This is the first team project, so students must form a team before this point. During the first semester of 2340, the UI milestone was required *before* the lecture on HCI design so that students could and would make mistakes to be later corrected. (Usability was a factor in grades on several milestones.)
3. Third assignment is non-programming: It's the complete plan for the project. We want to see OOA, OOD, as well as a detailed timeline and assignment of responsibilities to team members.
4. Fourth assignment is some serious team goal. During Spring 2000, this was the actual parsing of news sites.
5. Fifth assignment is the first of the two-part *twist*. Students should be told the details of this assignment around the time of the fourth milestone turn-in.
6. Sixth assignment is also related to the *twist*.
7. Seventh assignment is recovery from the *twist* and completion of the term project.

3 Updating the CoWeb

Much of the class information is exchanged (both ways) through the class CoWeb at <http://coweb.cc.gatech.edu/cs2340>. The goal is to use the same CoWeb term-after-term so that the resources (and past examples) might build-up over time and be available to future students. Here's what's involved in readying the CoWeb for a new term.

1. Make a class page somewhere for the old class, e.g. *Spring 2000 CS2340 Class Page*.
2. Copy-paste the old *Class Administration Page* into the old class page.
3. Now, edit the *Class Administration Page* to reflect the current class pages (milestones, project description, lecture schedule, etc.)

4. Similarly, create an old class *Who's Who* page (e.g., *Spring 2000 Who's Who*) and copy-past the current *Who's Who* content into it. Now, empty the current *Who's Who* page so that it's open for the next class to enter their info into it.
5. Be sure to update the *Hotspots* page to point to the *Readme First* and *Who's Who* pages from the very first day of class.
6. You may want to move and empty others, like the movie, restaurant, and book review pages in the *Personals* page, to encourage new ones. Or, you might want to leave them alone, to accrete new material.