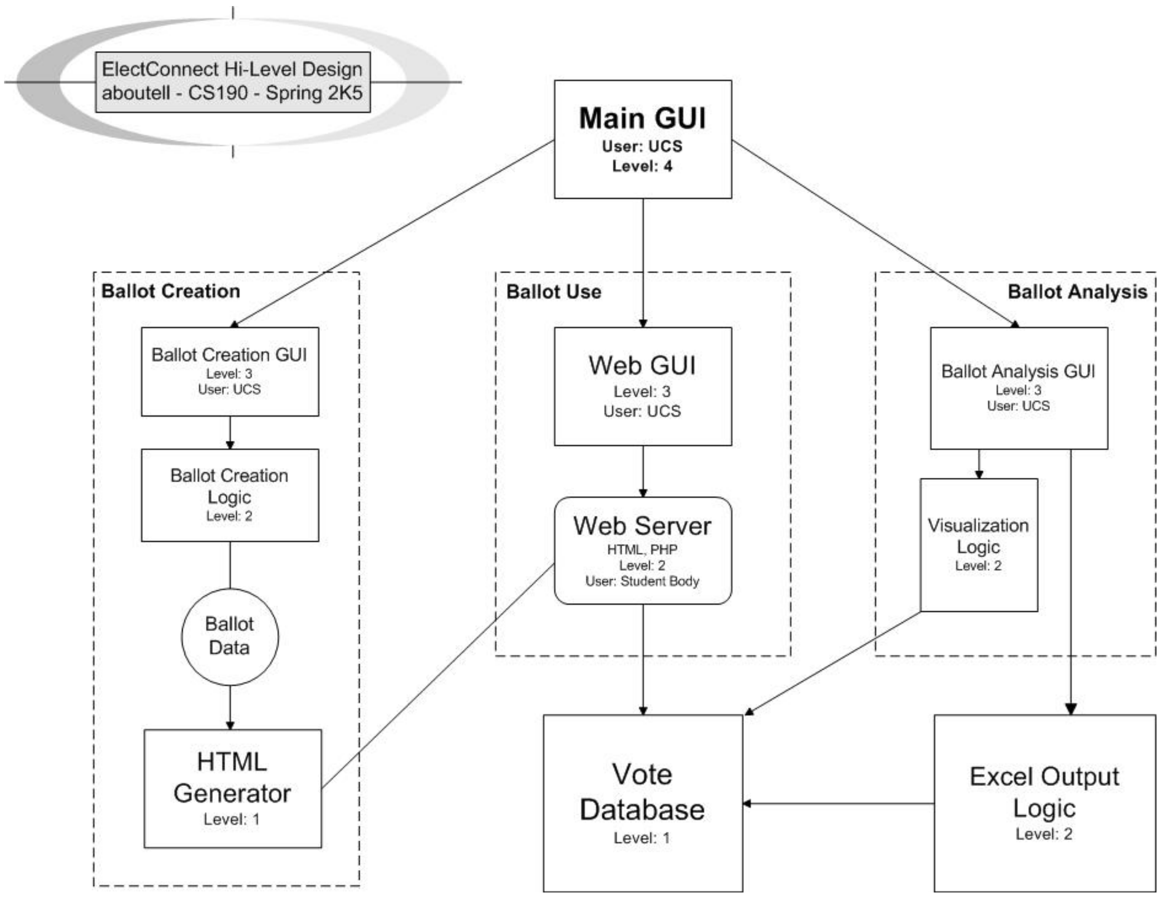
Alexander Boutelle
CS190 – Spring 2005 – aboutell

Hi-Level Design: ElectConnect

I. Overview

UCS runs a variety of elections and polls, and needs a robust, user-friendly online voting application. ElectConnect will be responsible for creating online polls, administering them to a 6000+ person audience, and analyzing the results (with output to standard spreadsheet format). The biggest drives behind our efforts need to be usability, reliability, and security. If the design is not considerably better than WebCT's services, our work is not an added value to the University. If the voting software breaks down or is compromised by a hacker, this application is a detriment. ElectConnect has great potential to improve the ease of student government, if implemented correctly.

II. HI-Level Design



Description of Components

In an effort to divide the work into specific categories, I tried to separate the general areas of Ballot Creation, Use, and Analysis (or as I prefer to think of it, Before, During, and

After). All three will be driven off of a single application that will be stored in the UCS servers. From a general opening screen, UCS users will be able to choose which function they desire, and from there use one of three GUIs (which will be unified in look and feel). To review the major sections of the application:

Ballot Creation GUI/Backend
This GUI will drive the ballot creation logic to design and create ballots that will be stored and run from the UCS server. The front-end will offer a variety of choices for ballot questions and answers, and will resemble the Wizard-style interface that users are comfortable with. A series of prepared steps will move them through the ballot creation process. The GUI will call method on the logical backend, which will prepare Ballot Data files for the HTML Generator. These Ballot Datas will contain all the content and formatting information for the creation of the ballot, and will probably be written in XML format.

HTML Generator
This class will create the HTML and PHP files that will comprise the online voting webpage. Ballot Data files from the Creation backend will be parsed and web files created from their specifications. These webpages will be stored in the online server. This does not count as a dependency, because the files will be stored and access independently (i.e. the HTML generator is a Level 1 because if we can get it to make webpages, we can view/test/access that independently).

Ballot Use GUI/Backend
The Ballot Use GUI will monitor the current status of created ballots, and which ballots are accessible online. It will also display summary information about the length of the voting period and how many users had already voted. The Backend will be thin, because we mainly need to display simple information and send commands (such as “Being Voting” and “End Voting”). The Ballot Use GUI will interface with the web server, which will function as a storage area for the actual ballot code, and a logistical bridge to the vote database.

Vote Database
A simple, but important piece of the puzzle. The vote database must be bulletproof in its reliability, and will be access regularly by the Ballot Use and Ballot Analysis groups. Its interfaces must be carefully defined and agreed upon, because any inconsistency will cause large and noticeable problems.

Ballot Analysis
This sector of the application will handle viewing and analyzing the results of an election. In some ways this is a flashy and non-crucial sector of the app – that is, it is not ***absolutely*** necessary for a functional and useful application. UCS will not release its results directly to the public, but will use a report with simplified and summarized Excel reports. For this reason, the absolute crucial side of Analysis section is the output to Excel format, which should be handled by the Analysis team as well.

III. External Dependencies

Security
This is one of our most important issues. We will need a way to authenticate individual users, to stop multiple votes. We will also need to encode (or otherwise protect) the information collected. There are a lot of very smart people at this University, and a lot of people who would sorely like to become UCS president, by any means necessary. This is a serious liability, and one we'll need to work with UCS and CIS to solve.

Hardware
We need to find out where and how this application will be stored and run. A server in UCS would be optimal for our needs, but might not be possible for budget/security reasons. This will need to be handled by working with UCS and UFB.

IV. Group Organisation

A. Group Leader

In charge of organizing and coordinating the group, of scheduling and running design and review meetings, and making sure that the work is being completed in parallel and on schedule.

B. Tester

The reliability and robustness of this assignment is crucial. Actual changes in Student government will be decided with this application, and so it needs to be bulletproof. Testing should occur as each phase of the app is implemented and integrated, and the tester should also be closely involved in the defining and agreeing of interfaces.

C. Project Librarian

This is not, repeat, <u>not</u> a puff job. We are designing software to live beyond our time at Brown, that will need to flexible, extensible, and very very usable. The documentation for ElectConnect needs to be industry-quality, the design should be accessible to a maintenance coder, and the code itself should be commented to the point that laypersons can understand our algorithms and functions. UCS members will need to be able to examine the code and assure its legitimacy. This project member will do less coding than other members, but may well have more actual work.

D. Coders (4)

The ground troops of the software industry, the coders will be involved in the design of our application, and actually go out and implement the assignment. One head coder should be oversee the interfaces and connections between the three sections, and then the three remaining coders should each take one section. The head coder will still be involved deeply with the creation and actual coding (as will the Project Leader, to a lesser extent), and will fill in wherever extra help is needed (i.e. the largest subsection of the project).

V. Schedule

- March 11 – Finalise top level design. I'd like to get this done ASAP, and believe we are within realistic shooting distance of this.
- March 16 – Basic Interface methods written out and traded for review
- March 18 – Final interfaces agreed upon – this includes storage file types/fields
- March 23 – Interfaces implemented and testing plan organized.
- April 6 – Detailed low level designs due. I'd like to make it quicker, but Spring Break is squarely in the way. After this point, we are in the depth of coding.
- April 15 – Integration between GUI and Logic begins
- April 20 – Systemwide integration begins. I budget less than a week for GUI/Logic integration because it will probably be handled by each coder independently, and that student will have an in depth knowledge of their sector of code. Systemwide will create a variety of new and interesting bugs and dependencies.
- April 22-24 - Spring Weekend. I will happily work through Spring Break, but I will not miss my Senior Spring Weekend. Those 3 days of the semester are off limits.
- April 27 – Post integration testing. This is of course a misnomer, because we will be testing continuously as we integrate, but by April 27 we should be in full on testing. I would schedule this
- April 29 – Feature Freeze – In Class Demo
- May 4 – Sunlab Demo
- May 16 – Final complete handin

VI. Assumptions

Security
This is the great elephant in the corner. We will need to design a full security plan (everything from password/login system on the actual box, to firewalls and online protection), and have it ready for the finalized top level design

File Format
The developing ideas about storing ballot and database data on static, independent files has a lot of potential, and not a lot of actual specifics. We are assuming we can develop these ideas, but it needs to happen by the time interfaces have been fleshed out.

Necessity
I want a piece of UCS stationary, signed by Joel Payne, that says they want this and will use it if we spend the time and energy to implement. I'm sure this is the case, but I want it transparent and assured.

Hardware
I am sure that UCS can supply us with a box to run this app off of, but we are not sure if that is the best solution. Currently, we'll assume that our app will run on some box somewhere at sometime.