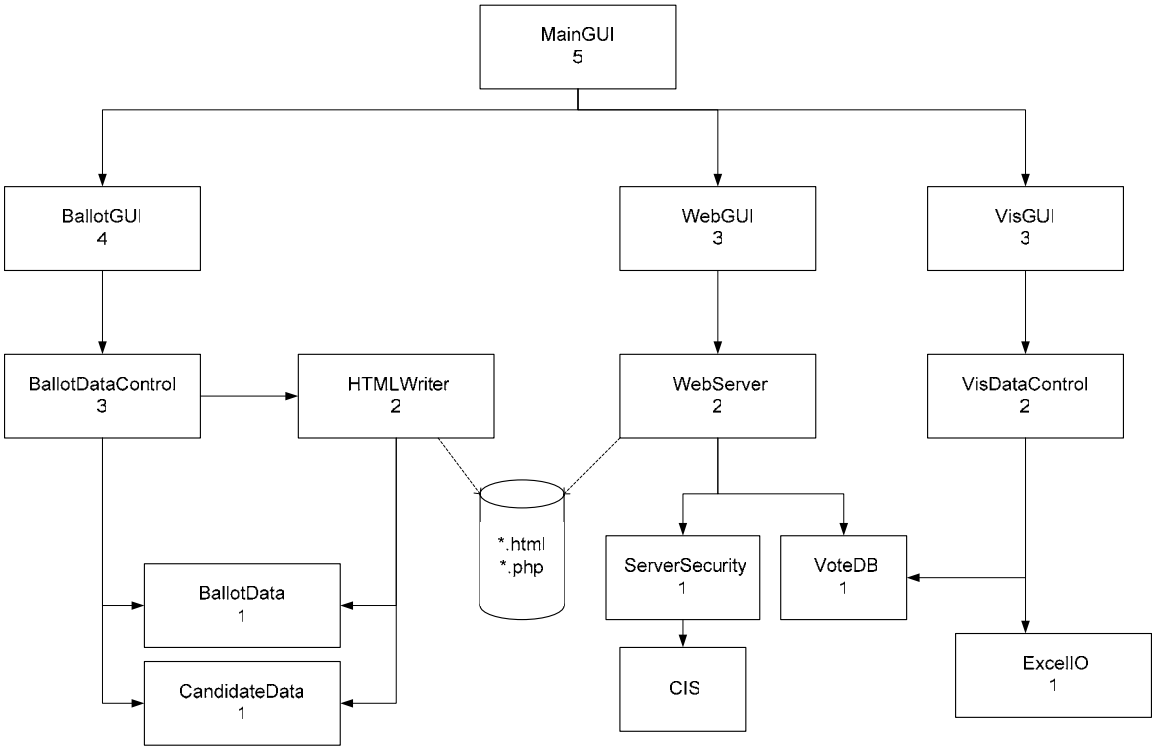
ElectConnect: Top Level Design

1.0 Description

The goal of this project is to design a system which allows the UCS Election Committee to design a ballot through the use of a straight-forward graphical user interface, similar to a standard Microsoft program. The software will then generate the necessary HTML, php and web scripts to administer the election online. The software will run a web server which students can connect to and submit their votes. At the conclusion of the election the system will end voting and will prepare a summary of the votes to be reviewed and analyzed by UCS.

2.0 High-Level Components

2.1 High-Level Component Diagram



Notes about the high-level component diagram

In this diagram the dark arrows indicate dependencies. In these cases, if component A has an arrow pointing to component B it is known that A will call methods on B and will depend on its functionality. The dashed arrows are meant to indicate that the HTMLWriter and the WebServer will both use .html and .php file which are written to disk. Finally, note that the CIS component is not included in the levelization. It is left

out because this represents an external dependency on the CIS login system. It is important to recognize this dependency in the top-level design, however it is assumed that this component is already fully implemented and working and thus will not affect the internal levelization of the project.

2.2 High-Level Component Description

As detailed in the specifications document, this project can be broken down in to three main modules: ballot design, election administration, and data analysis. The following discussion of the top-level components will address each module in turn.

2.2.1 Ballot Design Components

MainGUI

The MainGUI component is not discussed in the specification document and is not strictly a part of the Ballot Design module. It is a top level graphical user interface which allows the user to select what "mode" they want to operate in when they first run ElectConnect. It is modeled on CD burning applications like Nero and EasyCD Creator which, when first run, prompt the user with a screen asking them to choose whether they would like to create a data disk, a music CD, a CD from an image, etc. Based on the users selection, the program spawns a GUI unique to the task they are performing. Here the main GUI would operate similarly. The user will select whether she would like to create a ballot, administer an election or review election data. Based on their selection, the MainGUI will instantiate a BallotGUI, a WebGUI or a VisGUI. Since it was not presented in the specifications document, a screenshot of the MainGUI is provided here.

BallotGUI

The BallotGUI is the main graphical user interface for the ballot creation phase of the process and is similar in style and concept to the Microsoft PowerPoint interface. There is a standard system of Windows menus along the top of the screen. The File menu provides the user with functions to save and load files as well as to print and other standard functions. Similarly the Edit menu allows the user to make alterations to previously created slides. The Ballot menu will have options tailored toward the type of ballot they are designing, allowing them to add questions, add candidates and other election specific functions. Below the textual File menu will be a row of icons providing quick, easy access to common operations which are also available through the drop down menu systems.

The main section of the screen is divided in to two parts. On the left is a chain of thumbnails of the existing ballot questions. In the right hand frame are the specific parameters of the ballot question the user is currently designing. The user can select questions in the left hand frame and they will be brought in to focus in the right hand frame for editing.

The BallotGUI will call methods on the BallotDataControl component for two purposes, (1) to instruct the logic to update based on user input and (2) to query the logical component for information in order to keep the display up to date. An example of the first interaction would be the user clicking on the "Insert" menu and then choosing "New Ballot Question". This action would cause the GUI to call the `addBallotQuestion(.)` method on the BallotControlData class thus adding a new question with the user specified parameters to the election ballot. On the other hand, if the user wanted to edit an existing ballot question and clicked on the thumbnail of an existing question in the left hand pane of the BallotGUI, the GUI would call a method like `getBallotQuestion(.)` which returns ballot data to be displayed in the main portion of the screen. BallotGUI will interface entirely with the BallotDataControl.

BallotDataControl

BallotDataControl is responsible for the logical representation of the election ballot. Its primary method for accomplishing this task is the maintenance of the BallotData and the CandidiateData structures. BallotDataControl will be driven by user inputs through the BallotGUI and will determine whether or not the inputs are valid and, if they are, how to handle them. It is through this component that higher level user requests are turned in to a lower level representation of the ballot.

BallotControlData also contains an instance of the HTMLWriter. It would be possible for the HTMLWriter to be contained by the BallotGUI, however while this design would decrease the overall levelization of the top-level design this reduction would be in name only, would not actually lead to greatly reduced dependencies, and would be less consistent with the ideas for logical containment. In the case of the HTMLWriter,

BallotDataControl acts as an intermediary when the user clicks “Generate HTML” by passing along the `generate(.)` method call to the HTMLWriter.

BallotData

BallotData is a data structure which logically represents the questions on the ballot. The implementation of this structure is outside the scope of this document and should be handled by the engineer implementing this portion of the project. However, it is important to note that the design should be able to handle many types of questions (i.e. choose one of the following candidates, rank the following candidates, etc.) and should be extensible so that more question types can be added with minimal difficulty at a later date. Each item in the BallotData must represent a question type, the associated candidates, and whatever additional parameters the specific question type requires.

CandidateData

CandidateDate is very similar to BallotData. It is a data structure for storing information about the candidates eligible in the election. This information will be used to provide candidate biographies, pictures and personal statements on the election website.

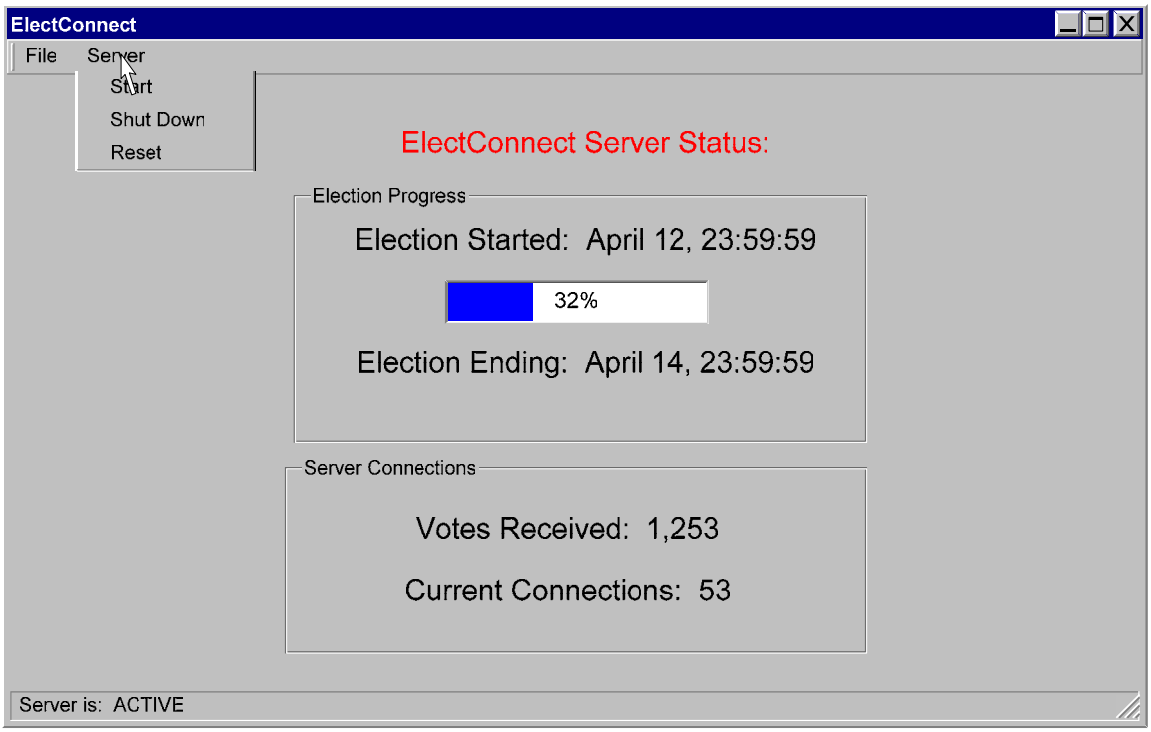
HTMLWriter

The HTMLWriter is responsible for generating html and php for administering the election website based on the information stored in BallotData and CandidateData. When the user directs the software to generate code for the election website, BallotData and CandidateData is passed to the HTMLWriter by the BallotDataControl. The HTMLWriter then iterates through the information held in each data structure and writes html and php to files.

2.2.2 Election Administration Components

WebGUI

The WebGUI is a graphical user interface used by the election administrator to monitor the status of the web server as the election is progressing. It may provide minimal functionality like the ability to end the election early or to change the time at which the election is scheduled to end, though this is not required. It must, at the least, provide a message regarding the server’s status and its current load. The intent is to provide confirmation that the server is running smoothly as the election is taking place. Again, because the WebGUI was not presented in the specifications document, a proposed design is below.



WebServer

The WebServer is the server users will connect to in order to view the election web page and to submit their ballots. It will host the site generated by the HTMLWriter. Server side scripting will also be provided via the php output by the HTMLWriter. The second function of the WebServer beyond serving the webpage is to take the data a user has submitted and to feed it in to the vote database where it will be stored until the election is over. The critical concern for the WebServer is its stability. Currently the goal is to support a total of 3000 submitted ballots along with the possibility of 200 simultaneous connections. However, these numbers may be revised upwards after additional conversations with UCS and CIS. The idea is to make these goals as aggressive as possible as the validity and the integrity of the election process depends in large part on the stability of the server. In addition to stability, security is a major issue in the administration of an online election. The WebServer will interact with a second component, ServerSecurity, to both verify user login information and to ensure that each user may only submit a single ballot.

ServerSecurity

ServerSecurity has two functions. First, it will interface with the CIS login system to provide a secure, well established and convenient method for recognizing users. Second, it will ensure that each user is only allowed to submit a single ballot.

VoteDB

The VoteDB is a database for storing aggregate voter responses to each question on the ballot. It will store raw data, unassociated with the user logins. For each question on the ballot, the database will store the number of votes that each response received. Information will be fed in to the database by the WebServer.

CIS

This object appears on the high-level design diagram to serve as a reminder of the dependency on the CIS login system as well as to help illustrate the role of the ServerSecurity component. It is fully implemented and maintained by CIS and we can assume it to have a well defined interface.

2.2.3 Data Analysis Components

VisGUI

The data visualization interface is a standard windows interface. It is intended for the group administering the election. It provides a framework for them to view and interpret the results of the election. At the very least it will have a drop down menu through which the user can select which ballot question they wish to analyze and a second drop down menu though which they can specify a visualization method (i.e. bar graph, pie chart, etc.). This interface will also offer options to view ballot questions which rank candidates as if they are instant run off questions. In addition the interface will allow the user the option of exporting the data to a spreadsheet.

It depends on VisDataControl to access the election result data. Based on the user's selections the VisGUI will request data from VisDataControl which it will then display in the appropriate format. If the users decides to export the vote data to a spreadsheet this message will be relayed to the VisDataControl which will export the data through the ExcelIO component.

VisDataControl

The VisDataControl component interacts primarily with the VoteDB to query for vote data which it then interprets manipulates and reformulates if necessary. VisDataControl is driven by user inputs to the VisGUI which are used to determine how VisDataControl should manipulate data. If, for example, the user is interpreting a question as an instant run-off question, she may choose to reallocate the votes of voters who choose the least popular candidate as their first choice. In this case VisDataControl will handle the redistribution and will return the updated data to the VisGUI for display.

ExcelIO

ExcelIO implements the critical task of writing the vote data to an excel formatted file. At the very least this functionality must be implemented in case the visualizer does not allow the user to perform all the required data manipulation. It will be important to have a working version of this portion of the data analysis module before dedicating significant time and resources to the VisGUI. ExcelIO will interface with the VisDataControl which can read vote information from the VoteDB and feed it to ExcelIO to be written out to the disk.

2.3 External Dependencies

2.3.1 Logins

At this point the major external dependencies are relatively clear. The first is the necessary use of the CIS NetID or AuthID login system. This will require working closely with CIS and may place some limitations on the functionality or the potential design options of our system. It will be important to understand how their system works before completing a detailed design of the WebServer and ServerSecurity components.

2.3.2 Server Hardware

A second external dependence is the required server hardware. While any windows based machine should be able to run the software to design the election and to view and interpret its data, however a more powerful (and stable) machine will be necessary in order to host the election website itself. We will have to give careful consideration to this issue as it will affect how we choose to approach the server software.

3.0 Group Organization

3.1 Project Manager

The Project Manager is effectively the group leader. As the leader, the Project Manager is responsible for maintaining the conceptual integrity of the software design and has the final word on design, interface and GUI decisions. However, it is critical that the Project Manager be comfortable and able to receive input and feedback from all members of the group. From an administrative perspective the Project Manager is responsible for maintaining a work schedule, enforcing deadlines and remaining informed of the general progress of the project. The Project Manager will also facilitate communication among group members and will be the arbiter of disputes should any arise. The Project Manager must also be available to advise on specific questions relating to code and to do research on questions put forward by the engineers.

Potential Project Managers:

- Xander Boutelle
- Lars Johansson

3.2 Ballot GUI Engineer

The Ballot GUI Engineer is responsible for the design and implementation of the Ballot Design GUI consistent with the requirements and goals laid out in the final project specifications. It is expected that this will be the most robust of the three user interfaces and thus a single programmer is being set aside for its implementation. This also means, though, that the Ballot GUI Engineer should have time to fully test their component at the pre and partial integration phases. Thus, the tester will have limited responsibilities when it comes to designing and trying test cases with respect to the Ballot GUI. The Ballot GUI Engineer should have an interest in the design of user interfaces and should be comfortable arranging third party user tests.

Also, the Ballot GUI Engineer will design and code the MainGUI, which should not pose a significant additional challenge.

Potential Ballot GUI Engineers:

- Michael Black
- Lars Johansson
- Xander Boutelle

3.3 Ballot Logic Engineer

The Ballot Logic Engineer is responsible for developing the low-level design and ultimately for implementing the BallotDataControl, BallotData, CandidateData and HTMLWriter components as described above. The Ballot Logic Engineer should be comfortable with data structures and with interacting with a user interface. Like the Ballot GUI Engineer, the Ballot Logic Engineer will also be expected to undertake almost all of the testing of their portion of the project. The Ballot Logic may not be the same volume of code as the other engineers are expected to produce, thus the Ballot Logic Engineer should have more time available to test their component both in the pre and post integration phases.

Potential Ballot Logic Engineers:

- Dan Silverman
- Alex Kossey

3.4 Server Engineer

The Server Engineer is responsible for the design and implementation of three components: the WebServer, the VoteDB and ServerSecurity. These components comprise the election administration module. The server engineer should have existing knowledge of web server programming as well as knowledge of or an interest in database programming. The Server Engineer will also have to spend time learning about the CIS login system and should be prepared for this. Because the programming here is expected to be complicated not only in terms of lines of code but also in terms of the diversity of areas the programmer will need to understand, the Server Engineer can expect the help of Project Support especially in the testing phases of their work.

Potential Server Engineers:
- Pawel Wrotek
- Lars Johansson

3.5 Visualization GUI Engineer

The role of the Visualization GUI Engineer is very similar to the role of the Ballot GUI Engineer. He is responsible for the design and implementation of the User Interface for data analysis and visualization. The data visualization module is the most flexible in terms of size and scope, so the Visualization GUI Engineer will be expected to work closely with the Visualization Logic Engineer, especially during the design phase, to fully detail and prioritize the requirements for this portion of the software. Also, the Visualization GUI Engineer should have an interest in developing creative and clear methods for displaying and interpreting data. The Visualization GUI Engineer will also be primarily responsible for the testing of this component, though help may be provided by Project Support.

The WebGUI will also be the responsibility of the Visualization GUI Engineer. This may seem unusual at first, however it makes sense that a team member with experience programming user interfaces should be assigned to the WebGUI. This is especially true because the WebGUI will be fairly small both in terms of features and lines of code. Thus, it would be far more efficient to have the Visualization GUI Engineer, who is already learning how to code a GUI, implement this as well than to require the Server Engineer to learn how to create a GUI on top of server and database code.

Potential Visualization GUI Engineers:
- Michael Black
- Catherine Hill
- Xander Boutelle

3.6 Visualization Logic Engineer

The Visualization Logic Engineer will work closely with the Visualization GUI Engineer to prioritize the features they will implement in terms of data visualization. She will be

responsible for designing and implementing the VisualizationDataControl and the ExcelIO components. The Visualization Logic Engineer should be proficient and have an interest in data manipulation and organization. They should also be knowledgeable about file I/O and have an interest in learning the specifics of Microsoft Excel file structure.

Potential Visualization Logic Engineers:
- Catherine Hill
- Dan Silverman
- Alex Kossey

3.7 Project Support

Project Support will be responsible for a variety of tasks. The three main functions Project Support will perform are (1) tool development and maintenance, (2) testing and (3) preparation of system documentation. Tool development and maintenance will most likely be a front heavy responsibility. It will include, among other things, setting up a logical directory structure, installing and configuring CVS, writing Makefiles and writing useful scripts. As a tester, Project Support will independently test individual portions of code and will assist especially in partial and post integration testing. This will provide a useful "reality check" for the other programmers since it is often best to have an outsider test your code because they do not necessarily understand how you intend the software to be used. Finally, documentation is an important part of this project since UCS will need to understand how to use the software and must be able to quickly and efficiently pass this knowledge along to future members. Thus, Project Support will be responsible for writing tutorials and help documents to accompany the software as a finished product.

The goal is that Project Support will remain busy over the course of the project as tool development will take place mostly at the beginning of the project, testing will be most helpful in the middle as the system engineers are still dedicating most of their time to writing new code and documentation will be prepared mostly at the end of the process when the attainable feature set is largely known.

Potential Project Support:
- Pawel Wrotek
- Xander Boutelle
- Lars Johansson

4.0 Schedule

March						
Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
		1	2	3	4	5
6	7	8	9	10	11 Final Design Doc. Complete	12
13	14	15	16 Initial Interface Design	17	18	19
20	21	22	23	24	25 Final Interfaces Complete	26
27 Spring Break	28 Spring Break	29 Spring Break	30 Spring Break	31 Spring Break		

April						
Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
					1 Spring Break	2 Spring Break
3	4	5	6 Detailed Component Designs	7	8	9
10	11	12	13	14	15	16 GUI/Logic Integrations
17 Early user GUI tests	18	19	20	21	22	23 System-wide Integration
24 Begin Post - Integration testing	25	26	27	28	29	30

May						
Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
1 Feature Freeze	2	3	4 SunLab Demo	5 Begin final testing and debugging	6	7
8	9	10	11	12	13 Docs. Complete	14
15	16 Final Handin	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

5.0 Updates

5.1 Assumptions

GUI Interaction

The specifications document detailed the difference between the three GUI's ElectConnect requires. However, it was not explicit about how they will interact, what binds them together, or how the user will navigate between the GUI's. This document clarifies those points by adding and discussing the MainGUI which occupies a place above the three second tier user interfaces.

Server Security

Server security and authentication were topics not covered in great detail in the specifications document. Hopefully by including the ServerSecurity component here the issue has been disambiguated or at the very least is now more clearly presented as one of the challenges of this project.

Component Interaction

The specifications document gave, I think, relevant explanations of how each of the high-level parts of ElectConnect are supposed to work, at least in this implementation. However, the interactions, dependencies and data flow among the various parts may still have been unclear. The goal then in this design document was to nail down how data flows between high-level components, where the dependencies lie and how the high-level system components interact with each other.

5.2 Updated Requirements

5.2.1 Highest

- provide an interface similar to standard Windows applications for designing the ballot on a question by question basis (PowerPoint as a model)
- generate HTML, scripts, etc. necessary to run a web-based election
- election website is clean and uncluttered with a well defined flow for the user to follow
- output election results data in excel ready format
- secure login system to ensure privacy and that each user votes only once

5.2.2 High

- host a site dedicated to administering the election
- the system must be able to support between 2,500 and 5,500 votes cast and a minimum of 200 simultaneous users
- support multiple election (referendum, survey) formats
- provide flexibility for different types of ballot questions (eg. vote for one of the following, rank the following candidates, etc.)
- use a database of users to provide different ballots to different logins based on class year

5.2.3 Medium

- provide a feature rich interface for analyzing and interpreting election results
- allow for a section of the election website containing candidate biographies and personal statements
- support for instant run-off voting
- web interface viewable through IE, Netscape and Firefox

5.2.4 Low

- support multiple election styles beyond majority wins and instant run-off

5.2.5 Lowest

- provide an interface through which the group designing the election can modify the appearance of the web ballot itself