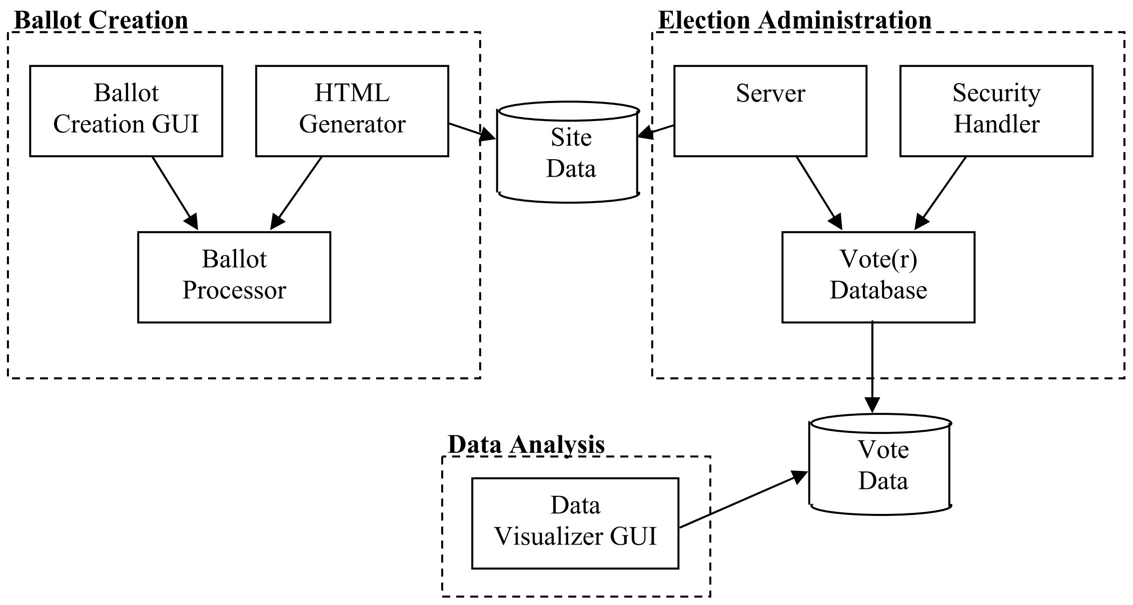
Election Connection **Top-Level Design** **Pawel Wrotek**

1. Overview

(from Lars Johansson's specification document)
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 would run a web server which students would 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. System Model Diagram



3. System Model Description

Ballot Creation GUI
This is the GUI that the user is presented with when creating the ballot. It allows him to add/edit/remove questions, specify the format of the election, load/save the ballot, etc. According to the specifications, the interface should be similar to Microsoft PowerPoint.

HTML Generator
This generates the HTML code and any necessary scripts for the election website, once the user specifies that he wants to create the site from the GUI. It outputs the Site Data that can then be used by the Server.

Ballot Processor
This is responsible for handling the data flow between the GUI and the HTML Generator. Whenever the user makes a change using the GUI, the GUI calls a method in the Ballot Processor, passing the updated data. When the HTML Generator wants to create the Site Data, it calls the Battle Processor, requesting the latest ballot data.

Server
This takes the Site Data produced by the HTML Generator and makes the election website accessible online. It collects the votes from the site, and passes them to the Vote Database.

Vote(r) Database
This receives and manages information about the votes that were cast and the users who have already voted. It also holds information about all the eligible voters (their user names and passwords) which is used by the Security Handler. This information can be input/modified by an administrator before each election.

Security Handler
When users first attempt to log in to vote, this module verifies their user names and passwords, and makes sure that they haven't already voted. If everything checks out, the user is given access to the voting site.

Data Visualizer GUI
This module works with the Vote Data taken from the Vote(r) Database. The election administrators use the GUI to display information about the outcome of the election in chart/graph format.

4. External Dependencies

- More information from UCS about what they want from the GUIs, or the project in general.
- A good user interface library (qt?).
- Security requires the use of CIS NetIDs. In the system presented above, these IDs and passwords would have to be acquired and fed into the Voter Database. This could be a huge stumbling block, since these probably aren't released even to the UCS election committee. Alternatively, we could rely on the CIS login system to handle most of the security (we would still need to keep track of whether the particular voter has already voted), but this would affect how useful our application is to the world outside of Brown.
- A computer to run the server, and web space for the voting site (though this should be handled by UCS or whoever runs an election using our program).

5. Task Breakdown/Group Organization

Project Manager/Ballot Processor - ljohanss
This person is responsible for making sure that all parts of the project are on track. Also, he is our connection with UCS and must make sure that the specifics of the project (especially GUI and website functionality and layout) correspond with what UCS wants. This person is also responsible for the Ballot Processor module – the smallest (quite possibly trivial) part of the project.

Tester - xb
This person is responsible for creating scripts to rigorously test the individual components, as well as the combined product. He should be well versed in what is expected from each component, what kind of input it might receive, and what kind of output it should produce. While everyone needs to test their own code to find major bugs, the tester should design tests that also look for subtler errors and problems that occur after integration. The tester is also responsible for writing the documentation.

Ballot Creation GUI – miblack
This person should work closely with the person responsible for the Data Visualizer GUI to make sure that there is some sort of coherence between the two GUIs.

Data Visualizer GUI – pwrotek
This person should work with the person responsible for the Ballot Creation GUI (see above).

HTML Generator - akossey

Server - cjhill

Security Handler/Vote(r) Database - desilver
If it turns out that we must let CIS handle most of our security needs, than the Security Handler will be reduced to simply checking whether a user has voted before. The Voter Database component is closely related to the Security Handler, and should not be a significant amount of work.

6. Schedule

3/4	Initial group meetings; discuss/choose top-level design
3/11	Overall project design in; top-level design frozen
3/16	Interface proposals
3/21	Project manager decides final list of features together with UCS representative
3/25	Interfaces finalized; begin coding; Tester begins creating test suites for individual components
4/4	Project manager and Tester revises design, makes final revisions, decide general integration/testing plan
4/8	Detailed designs due
4/18	Initial integration; testing suites for individual components finalize, made available to coders; Tester begins writing documentation

4/29	Full integration; functionality freeze; in-class demo; from now on lots of testing/debugging, NO NEW FEATURES added
5/16	Final demos; final documentation, code, testing reports

7. Assumptions

The specification document mentions the need for security, but nothing is said about how this will be handled by the system. I assumed that the system will have some way of handling it (the Security Handler module) and that we will not simply rely on CIS.