Software Requirements Specification

for

Voting System

Version 1.0 approved

Prepared by Alex Bohm, Peter Linden, Andrew Petrescu, Nikhil Srikanth

CSCI 5801 Team 20

02/02/2021

Table of Contents

Revision History

Name	Date	Reason For Changes	Version

1. Introduction

1.1 Purpose

The purpose of this document is to specify the software requirements for the Voting System, a system that counts ballot results and determines a winner based on specific vote counting methods.

1.2 Document Conventions

This document follows the IEEE template for Software Requirement Specification documents. Regular fonts are used for the content and larger, bolded fonts are used for heading and subheadings. Numbered lists are used to show sequences of events. Every requirement statement has its own priority, statements that are functionally related have similar priorities.

1.3 Intended Audience and Reading Suggestions

The intended audience for this document includes election officials or whomever else may be a user of the Voting System, testers who need to ensure the application functions as intended, and other engineers/developers who wish to work on this application. All users should start with the introduction and overall description of the application. Audience members who will interact with the Voting System in the development and testing phase should continue reading the document starting with the next subsection. Users like election officials who will interact with the Voting System in its finished state should be able to skip section 2 and start reading again at section 3.

1.4 Product Scope

This Voting System is an application that is intended to make the process of counting votes and determining the results of an election much easier. The Voting System is intended to be able to process and generate results for two different election types: Instant Runoff (IR) and Open Party Listing (OPL). After the user provides the system with a ballot sheet, the system is able to determine the type of election, count votes according to the rules of the election type, create an audit document, and determine and display the results both to the screen and exported to a media text file.

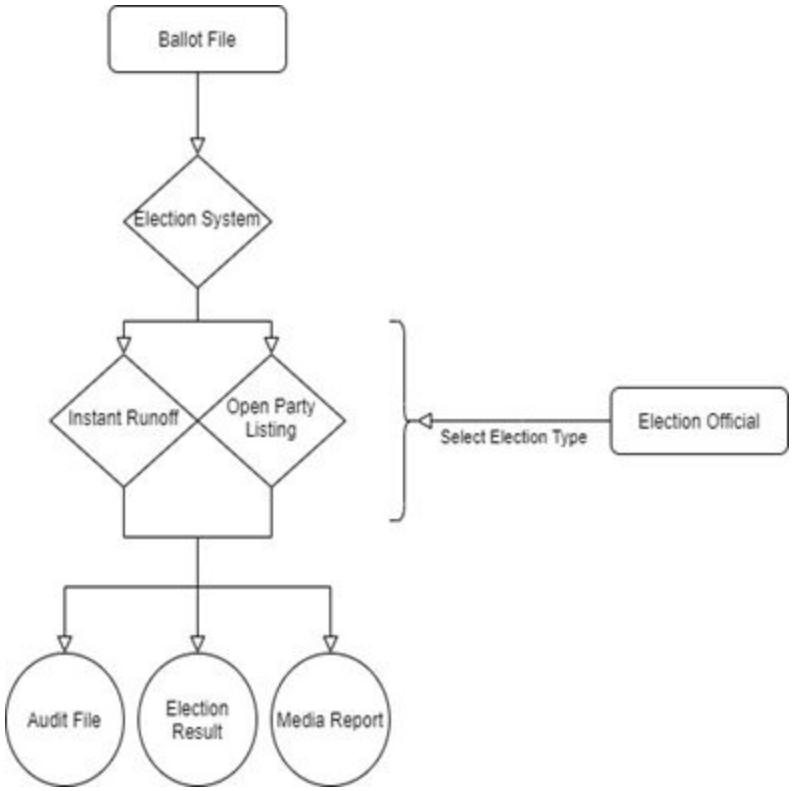
1.5 References

Initial Requirements:
https://canvas.umn.edu/courses/217849/files/18790396?wrap=1

IEEE SRS Template:
https://canvas.umn.edu/courses/217849/files/18690407?wrap=1

2. Overall Description

2.1 Product Perspective



The Voting System is developed with the intention of providing a software method of determining the results of chosen elections in a logical, efficient, and transparent fashion. It is not built off any existing systems - it is intended to stand alone as an election software solution. It fulfills the needs of all the actors involved: election officials get an unconvoluted, digital method of counting election results, the media get auto-generated statistics and metrics to report on, and audit officials get a transparent and organized view of how the election software was run.

2.2 Product Functions

- *Run program – Loads up software and begins prompting user for ballot data and type*
- *Select Ballot File – User identifies ballot file to be used*
- *Read ballot file – System imports ballot file from election needed to be counted*
- *Label Ballots – System goes through ballot file and gives each ballot an ID to use in counting methods*
- *Indicate election type – User selects between running Instant Runoff or Open Party Listing methods on the election data*
    - *Determine winner in IR – System counts votes using IR method*

    - *Group candidates by party, including independents – System groups candidates by party in order to properly allocate votes*
        - *Determine winner in OPL – System counts votes using OPL method*

- *Display winner/results to screen – System displays the winners of the election to the screen*
- *Show redistribution votes of IR in audit file – System recreates flow of logic that it used to reallocate votes in the IR method*
- *Produce Audit file – System creates audit file containing all counted ballot IDs, and how they were distributed to candidates at all stages of the election*
- *Resolve Tie – In the event of a tie, the system performs a random coin flip to determine the winning candidate / party*
- *Produce report for media personnel – System generates report containing election metrics like vote totals, candidate / party performance and other things to report on*
- *Retrieve audit file – Audit official queries the system to export the audit file to a viewable format*

2.3 User Classes and Characteristics

- Election officials, who oversee running elections and overseeing the method of counting the votes properly. They are authorized to interact with the ballot data, and run the election software, and are thus the most crucial user class for this product
- Audit officials oversee ensuring that the election was carried out correctly and will thus use the system to get a log of how ballots were counted and distributed. These officials are authorized to interact with the system, but only to generate the audit log, not to count votes. These actors are less likely to interact with the system every election like the media or election officials, but their role in ensuring election transparency following a disputed result means that they are much more important than their frequency of interaction would suggest
- Media officials interact with the system indirectly, in that they are sent reports containing the statistics about the election results. These reports are auto-generated and sent from the system to the media, so the media do not have authorization to utilize the system
- Testers interact with the system to ensure that all the features operate as expected. This includes testing the behavior of the system in response to new election data, running the software on the two different election styles, and making sure that the proper outputs are presented. The tester is also an important user, because they help ensure that the product is as functional as can be
- While not a typical user class, the system itself is considered an actor for several of the proposed features for the Voting System. The system should expect to execute certain functions given the inputs from the election official, and will continue propagating the results to the actors at the output of the Voting System.The system is ranked as a high importance user due to its involvement in every step in the election process.

2.4 Operating Environment

The recommended machine for any user (Election Official, Tester, etc.) of the program is a UMN CSE IT Lab machine running a Linux based operating system. These machines are found in Keller Hall (rooms 4-250, 1-250) and are running Ubuntu 20.04. They are built from Dell Optiplex 9020 desktop computers and are upgraded to contain 32 GB of RAM and an Intel Core i7 processor.

The CSE IT Lab Machines also include Windows 10 and Mac OS X in some buildings, but those are not recommended for this project as system features are written with a Linux environment in mind.

For the development of this product, other machines were leveraged to make the process smoother. These personal computers run on Windows 10 and Linux with varying processors and RAM.

2.5 Design and Implementation Constraints

Developers must expect the final software product to work within the specs of the UMN CSE IT Lab machines, the specific requirements for which are described in the previous section, 2.4.

The software is written in the programming language of C++ and is made solely for those who know the English language. The RAM used for this project is capped at 32GB which should be more than enough to satisfy the given timing requirements, which is to be able to run 100,000 ballots in under 8 minutes. The user of the Voting System must have read, write, and execute permissions for the associated files. There are no other security checkpoints besides the file permissions.

Once delivered, the original developers will not be responsible for maintaining the software.

2.6 User Documentation

A detailed README instruction file will be included with the delivery of the final product.

2.7 Assumptions and Dependencies

It is assumed that any user of this program has a basic working knowledge of linux and how to navigate/execute files using the command line. Any user who wishes to use this program must have access to the UMN CSE IT Lab machines located in Keller Hall, whether it be remotely or in person.

Other assumptions include those given by the requirements document:

- The ballot file will be in CSV format
- The ballot file will be in the same directory as the program executable
- The Voting System will not be run on the same ballot file twice
- The ballot file has no mistakes, and each person votes only once

- Users who use the program have the required authority to do so
- The most up-to-date CSELabs machines will be used

3. External Interface Requirements

3.1 User Interfaces

The user interface for the Voting System is a simplistic, command line based interface. Standard usage will have the user run the executable with arguments that specify the ballot file.

```
./voting-system <filename>
```

Further arguments can be applied to the program using standard unix command line arguments.

```
./voting-system --test <filename>
```

Future interfaces that may be applied to the software include prompts for entry of election type, however it is not expected to be used in the initial implementation of the software as the ballot files already contain the election type.

The interface will print confirmation of the election type, periodic messages as the counting occurs, and finally the results of the election. For the specific types of vote counting, short messages describing the major use cases such as vote redistribution shall be printed.

The final results of the election shall show the winners, losers, and vote statistics.

3.2 Hardware Interfaces

The Voting System runs on any machine that has a readable and writable filesystem attached. The program must be able to read the ballots from a file and must be able to write to an audit log and a media release file. While optional, it is highly recommended that this program be run from a console as election officials should monitor the counting process. For machine specific requirements refer to section 2.4, Operating Environment.

3.3 Software Interfaces

This software interfaces heavily with the operating system to provide file input and output operations. Using the standard C++ iostream library, the Voting System reads and writes multiple files. The software relies on the operating system to handle low level file manipulation.

3.4 Communications Interfaces

This software is a standalone product that interfaces with other systems through the use of files. Once given a ballots file from another system, this system produces an audit log and media report that can be passed onto a different system.

4. System Features

4.1 SF_001 Run Program

4.1.1 Description and Priority

The election official navigates to the directory where the program is through the command line. Once in the desired directory, the official enters the appropriate linux command to run the program executable. This is a high priority feature.

4.1.2 Stimulus/Response Sequences

Trigger: An election has closed its polls and the final ballot tally is sent to the Voting System in order to determine the results.

Precondition: The election has finished and the user is a qualified election official who knows basic linux commands to navigate the terminal and run executables.

Postcondition: The Voting System begins its processes to determine the results of the election.

Sequence:

1. User logs into CSE-IT lab machine
2. User opens the terminal
3. User navigates to directory with program executable using linux commands
4. User enters linux command to run the executable

4.1.3 Functional Requirements

REQ-001: CSE-IT lab machine uses a linux operating system

- If on other machine, find and log into a linux based system

4.2 SF_002 Select Ballot File

4.2.1 Description and Priority

The election official must locate the ballot file in the program directory. This file will then be entered as a command line argument when running the program. This is a high priority feature.

4.2.2 Stimulus/Response Sequences

Trigger: An election official is about to run the vote counting program and is in the directory with the program and ballot file.

Precondition: The election has finished and the user is a qualified election official who is about to begin the process of running the program.

Postcondition: The vote counting system is able to read in the ballot file and count the votes to determine the results.

Sequence:

1. User is in directory that contains the ballot file and the program executable
2. User locates the ballot file and identifies its name
3. User inputs the name of the file as a command line parameter when running the program (During step 4 of 4.1.2)

4.2.3 Functional Requirements

REQ-002: Ballot file name is input correctly

- Program will fail to run if there are spelling errors, user must run again with correct name

4.3 SF_003 Read File

4.3.1 Description and Priority

The system will read in the ballot file that was input as a command line parameter. The major details required for the correct counting of votes will be extrapolated from the file and stored within the program as necessary. The vote counting process can then begin. This is a high priority feature.

4.3.2 Stimulus/Response Sequences

Trigger: An election official has entered the correct linux command to run the program with the ballot file input as a parameter.

Precondition: The vote counting system has been run from the command line with the correct file parameter.

Postcondition: The vote counting system extrapolates the details of the file and begins the vote counting process. The ballot file remains unchanged.

Sequence:

1. System is run with ballot file passed in as parameter
2. System opens file
3. System reads file line by line and stores data in necessary locations
4. System closes file

4.3.3 Functional Requirements

REQ-003: Voting System is able to open and read the file

1. If unable, detailed error messages will be displayed and the program will end.
    a. Error details will indicate point of failure
        i. Incorrect filename
        ii. Incorrect file type
        iii. Incorrect permissions
        iv. Etc.
2. The user will have to run the program again with adjustments made based on the error message.

4.4 SF_004: Indicating Election Type

4.4.1 Description and Priority

The user, likely an election official, will use system prompts to confirm which election type the given ballot file belongs to, either instant runoff or open party listing. This allows the election official to have final control over the way that the Voting System runs. On the software side, allowing for the selection between the two allows one program to fit the two election systems. This feature is utilized every time a ballot requires tabulating, i.e. for every use of the election system software, and thus has a high priority.

4.4.2 Stimulus/Response Sequences

Trigger: The election official starts the election system software and selects the ballot file using the command line interface. The election official then is prompted to choose the election type.

Precondition: The ballot file is loaded correctly.

Postcondition: A response confirming the election type is displayed on the screen. The election type in the ballot file matches the election official's response.

Sequence:

1. User is shown a prompt on the CLI giving two options for the election type, I (Instant Runoff) and O (Open Party)
2. User enters one of the characters to dictate their election type
3. User presses enter
4. System confirms the choice, and displays the choice
5. System prompts the user to confirm that it should go ahead with tallying the votes with a Y/N option
6. User presses Y and then enter, and election system software commences

4.4.3 Functional Requirements

REQ-004: Response to Election Type prompt is of acceptable format (I/O)

1. System prompts user a second time
1. If input is of correct format, continue to MC4
2. If input is again incorrect, display error message

4.5 SF_005: Determine winner in case of clear majority in Instant Runoff

4.5.1 Description and Priority

System determines the winner of an Instant Runoff election where a candidate earns over 50% of the first choice votes, those being made up of each ballot's first choice candidate, without eliminating and re-allocating less popular candidates. Like SF_004, this feature is used every time a ballot is loaded with the intent of counting the votes via the instant runoff election type. Thus, it is used every time an election official selects IR as the election type, and is of high priority.

4.5.2 Stimulus/Response Sequences

Trigger: The system records that the user confirms instant runoff as the chosen election type and commences with the counting of the ballots.

Precondition: Instant Runoff is the selected election type. A valid ballot file has been uploaded.

Postcondition: A candidate with >50% of the initial vote is determined to be the winner by the system. If there is no initial winner, the system continues the instant runoff methodology of selecting a winner.

Sequence:

1. Ballot file is opened
2. An empty data structure is created to hold all candidates and the IDs of each ballot that had said candidate as first choice, as well as said ballot's 2^{nd}, 3^{rd}, 4^{th}, and 5^{th} choice candidates. The 2^{nd} – 5^{th} choice elements will be used for re-allocation of ballots whose first choice was eliminated due to being the least popular in a particular voting round
3. Ballot results are read sequentially
    a. If the ballot result is for a candidate not yet in the data structure, create a new entry
    b. If the ballot result is already in data structure, increment the vote count
4. Finish ballot counting
5. If there is a candidate with >50% of the total votes, determine them the winner

4.5.3 Functional Requirements

REQ-005: The initial first choice vote count has given a candidate >50% of the vote
1. System moves on to the elimination/re-allocation of less popular candidate vote totals as enumerated in SF_006.

4.6 SF_006: Determine winner in case of no clear majority in Instant Runoff

4.6.1 Description and Priority

The system sequentially eliminates less popular candidates and redistribute votes to decide a winner in the event where no candidate gets more than 50% of the votes when Instant Runoff is chosen as the election type. This is also a high priority feature, as it is essential for proper allocation of IR votes.

4.6.2 Stimulus/Response Sequences

Trigger: The system records that no candidate has earned >50% of the initial votes, and thus no popular majority has been achieved.

Preconditions: All the ballots in the ballot file have been counted and have all been allocated to their candidate of choice.

Postconditions: A winner of the instant runoff election is determined.

Sequence

1. Candidate data structure is searched to find the candidate with the fewest votes
2. Go through the ballots attached to said candidate as their n choice, and add them to the ballot list for the candidate listed as the n+1 choice for each ballot
3. The winner is determined to be whoever received over 50% of the vote after the reallocation of the least popular candidate's ballots

4.6.3 Functional Requirements

REQ-006: The methodology must terminate with a candidate earning >50% of the vote total after accounting for re-allocated votes.
1. If this is not the case, the 3 steps in the sequence are simply run through again, starting at 1, re-allocating the next least popular candidate's votes.
2. In the event that the results are tied with two remaining candidates, move on to SF_013 to resolve tie

4.7 SF_007: Group Open Party Listing Candidates into respective parties

4.7.1 Description and Priority

Candidates of the same party must be categorized together for proper vote allocation. Independent candidates without an affiliated party are grouped together by the system so that the open party listing method of deciding election results can work. This feature allows votes for independents to be properly allocated, since they don't have an existing party affiliation that is used to categorize votes. It allows the system to follow a generalized structure of vote allocation without having edge cases relating to interacting with independents. This feature is utilized every time a ballot file is to

be tabulated using the open party listing methodology. The election in question may not have any independents to group, but the system will still check every time. The priority is high.

4.7.2 Stimulus/Response Sequences

Trigger: User of system has selected election type as OPL

Precondition: System able to read in ballot data.

Postcondition: All candidates are now grouped in a data structure, allowing the open party listing voting algorithm commence.

Sequence

1. System creates data structure where each element contains the candidates running under the same Party affiliation
2. System goes through ballots, and looks at candidate and party choice
3. If the party affiliation does not yet have a spot in the data structure, add one
4. If the candidate is not yet in their respective party's slot, add them
5. If the candidate is independent, add them to a slot reserved for independents.

4.7.3 Functional Requirements

REQ-007: Ballot file must have affiliated party for each candidate in the election

4.8 SF_008: Display Election Winner

4.8.1 Description and Priority

The system must be able to display election results. The results include the election winners and losers, along with common statistics about the votes. This feature is utilized at least once per run - the election official and tester interact with the feature such that they view the displayed output, and the system is the primary actor in that it does the displaying. This is a high priority feature.

4.8.2 Stimulus/Response Sequences

Trigger: All ballots have been counted and redistributed if necessary

Precondition: All ballots have been counted, redistributed, and ballot data structure changes have been completed.

Postcondition: The screen displays the results of the election and related statistics. The data structures representing the ballots have remained unchanged.

Sequence

1. Election results are requested
2. Data structure representing election type is interrogated for results (AC1)
3. Results are formatted and printed to the terminal screen

4.8.3 Functional Requirements

REQ-008: All ballots must be counted for results to be displayed
   1. If counting is still in progress, the screen should show current progress of program

REQ-009: Must be able to display:
- Number of ballots cast
- Winners
- Number of ballots received for each candidate
- Election type
- Number of seats

4.9 SF_009: Start Audit Log

4.9.1 Description and Priority

Create the audit file and write a header containing information pertinent to the instance of the program such as the time and date and the name of the ballots file. This feature is used once per audit log creation, and occurs once per program execution, making it a high priority feature. The system automatically will start an audit log during program execution.

4.9.2 Stimulus/Response Sequences

Trigger: An audit log is requested for the program

Precondition: The file system is in a writable state

Postcondition: An audit log with a header containing metadata about the election is in the current directory

Sequence

1. Generate file name based on current date and time
2. Check if file exists on filesystem
3. Create and open file
4. Write election type, time, and other metadata to file

4.9.3 Functional Requirements

REQ-010: Audit log naming must be unique
1. If the file already exists, follow numbering scheme and re-try file creation until successful
2. Simply increment filename until unique

REQ-011: Audit file must be writable
1. System must create file with writing permissions

REQ-012: Audit log must log the following:
- Type of Voting
- Number of Candidates
- Candidates

4.10 SF_010: Log Ballot Ingest

4.10.1 Description and Priority

During the initial ingestion of ballots, each ballot shall be logged to the audit log in order to produce a trace of the ballot through the system. The audit log is automatic to the system, therefore the only interaction necessary for this use case to be called is that the program runs and has some ballots. This feature is high priority as the system needs to record the reading and movement of ballots.

4.10.2 Stimulus/Response Sequences

Trigger: A new ballot has been read from the election ballots file

Precondition: A new ballot has been created and has not been logged yet

Postcondition: The logged ballot is unchanged and the audit log now contains an entry with all the information contained in the ballot.

Sequence:

1. A newly created ballot is passed to the audit log class
2. Information about the ballot is formatted into an audit log entry
3. The audit log entry is written to the audit log
4. Success flag is returned

4.10.3 Functional Requirements

REQ-013: Must be able to write to audit log
1. Error flag is returned if unable to write

REQ-014: Must log ballot and associated id

4.11 SF_011: Log Redistribution of Votes

4.11.1 Description and Priority

In Independent Runoff voting, the audit log should show the redistribution of votes. During a counting process just under 50 percent of the votes may need to be redistributed, this means upwards of 50,000 votes may need to be logged. As the audit log is an automatic response, the main actor for this use case is the system.

4.11.2 Stimulus/Response Sequences

Trigger: A ballot is being redistributed

Precondition: A ballot is being redistributed

Postcondition: The supplied ballot is unchanged, but the audit log now contains an entry showing that the ballot was moved to another candidate.

Sequence

1. A ballot for redistribution is passed to the audit log class along with the candidate the ballot is moving to.
2. Ballot information is formatted into an audit log entry with the candidate the vote is now counting for.
3. Audit log entry is written to the audit log
4. Return success flag

4.11.3 Functional Requirements

REQ-015: Must be able to write to audit log
1. Return error flag

REQ-016: Audit log must show redistribution of ballots

4.12 SF_012: Log Election Results

4.12.1 Description and Priority

Write entries to the audit log that show the results of the election, including the winners, losers, and ballot statistics (total count, percentage per candidate, etc). This use case may be run at least once per count, however as it is necessary to view partial counts, this use case should also output information about a count that is in progress. Typical application sees this use case as being called under 10 times during the counting process, however Auditors and Testers may need more frequent calls. The System and Election Official should only need this use case under typical usage with up to 10 usages.

4.12.2 Stimulus/Response Sequences

Trigger: Election results are requested to be logged.

Precondition: Data structures representing the type of election are ready to be interrogated.

Postcondition: Election results are logged to the audit file.

Sequence

1. Data structure representing the Voting System is interrogated for results (AC1)
2. Results are formatted into an audit log entry
3. Write audit log entry to audit log (EX1)
4. Return success flag

4.12.3 Functional Requirements

REQ-017: In progress results must be labeled with an 'In Progress' flag written to log

REQ-018: Must be able to write to audit file

1. Return error flag

REQ-019: Log must show

- Type of voting
- Number of candidates
- Candidates
- Number of ballots
- Number of ballots per candidate

4.13 SF_013: Resolving a Tie

4.13.1 Description and Priority

When the election does not end with one party having a majority vote, a coin flip will determine the winner of the election. To ensure fairness and randomness, the coin will be flipped 100 times and the 101’st flip will be the determining flip. This system function has a medium priority.

4.13.2 Stimulus/Response Sequences

Trigger: The election does not result with one party having a majority.

Precondition: Election ballots have been logged.

Postcondition: A winner for the election is announced.

Sequence:

1. Ballots have been logged and counted and no party has a majority
2. System will do 101 coin flips
3. System will read 101'st coin flip
4. Winner of the final coin flip will be announced as the winner of the election

4.13.3 Functional Requirements

REQ-020: Ballots have been read and logged into system

REQ-021: System has done calculations to determined that no party a majority

REQ-022: System can do coin flips to determine election winner

4.14 SF_014: Produce Report for Media Personnel

4.14.1 Description and Priority

When the election results have been logged and a winner has been determined, the media file will be released to the media as a text file. This system function has a low priority

4.14.2 Stimulus/Response Sequences

Trigger: Winner has been determined.

Precondition: Winner has been determined.

Postcondition: Media file is stored and ready to be sent to the media.

Sequence:

1. Log results have been completed
2. Log results are stored in a media text files

4.14.3 Functional Requirements

REQ-023: Log results have been finished

4.15 SF_015: Retrieve Audit File

4.15.1 Description and Priority

Takes the created audit file, which has the required information, and makes it visible to the auditor as a readable file. This system function has a high priority.

4.15.2 Stimulus/Response Sequences

Trigger: Audit file has been created and election results have been logged

Precondition: Audit file exists in the program directory.

Postcondition: Audit file is available to be read by the auditor.

Sequence:

1. Navigate to program directory

2. Locate audit file
3. Auditor accesses file

4.15.3 Functional Requirements

REQ-023: Audit file must exist

REQ-024: Logged results must be completed

REQ-025: Logged results must be added to the audit file

5. Other Nonfunctional Requirements

5.1 Performance Requirements

The Voting System will be able to run 100,000 ballots in under 8 minutes given the memory limits and computer power given in the Operating Environment section.

5.2 Safety Requirements

There are no foreseen safety requirements for the election software. The Voting System will satisfy all the IEEE standards in regards to safety requirements since none were listed in the initial document.

5.3 Security Requirements

While the Voting System is running, calculations and/or voter information should not be able to be altered. The user is responsible for the security of the machine they use.

5.4 Software Quality Attributes

Adaptability/flexibility: You can pass the name of the file into the program as a command-line argument. The software will contain a generic ballot parsing system that can supply data to an extensible ballot counting class.

Availability: Ballots will be cast online and a comma delimited text file will be provided.

Correctness/reliability: The Voting System will output the correct winner of the election with no mistakes.

Interoperability: Using the CSE lab machines to write to files and get information from the ballots.

Portability: The software will use C++ standard libraries (std). As long as a system contains implementations for the standard libraries the software should be able to be easily ported to said system.

Reusability: The Voting System must run multiple times during the year at normal election times and special elections and still produce the correct winner(s).

Robustness: The Voting System takens in few inputs, making it resistant to external errors. It also has robust exception handling to deal with unexpected operations.

Testability: The Voting System needs to produce an audit file with the election information at the time, it should list the winner(s), and it should show how the election progressed so that the audit could replicate the election itself. It should also show how it got what ballot and its order of being received if applicable.

Usability: The system takes a single file name and runs to completion without further input on a normal ballot file.

5.5 Business Rules

Special access will be granted to election officials and auditors. The access for election officials would allow for one of them to start the vote counting process. The access for auditors would allow them to only view audit reports from the election once completed. The media will be eligible to receive a report formatted for the information they need if requested. No other access will be granted to any other individual and no access will allow for altering of election ballots to maintain fairness of the election.

6. Other Requirements

Appendix A: Glossary

Independent Runoff Voting (IR)
Open Party Listing (OPL)
Command Line Interface (CLI)

Appendix B: Analysis Models

TBD

Appendix C: To Be Determined List

See Appendix B