CSci 5801: Software Engineering I, Spring 2018
Project 1 – Waterfall Methodology
Software Requirements Specification (SRS) Document for Voting System
Due Date: Sunday, February 11 at 11:55 p.m.
100 points total

Special Instructions: You will be working in your small groups to complete this homework assignment. You should meet, skype, or talk on the phone (if unable to meet in person) about the requirements for the assignment. You will only turn in one assignment per group. You must include all names on your assignment with X500 names. Please use the name that is listed on the class roster so we will know who you are. You will upload your work to Moodle. Only one person should upload to Moodle. We expect you to turn in your Software Requirements Specification document (must be typed) and your use cases (either incorporated in your SRS document or as a separate document (must be typed). You will be allowed to upload multiple documents. Be aware that you will not be able to upload after 11:55 p.m. on Sunday, February 11th. If you are not done, turn in what you have so we can give you partial credit.

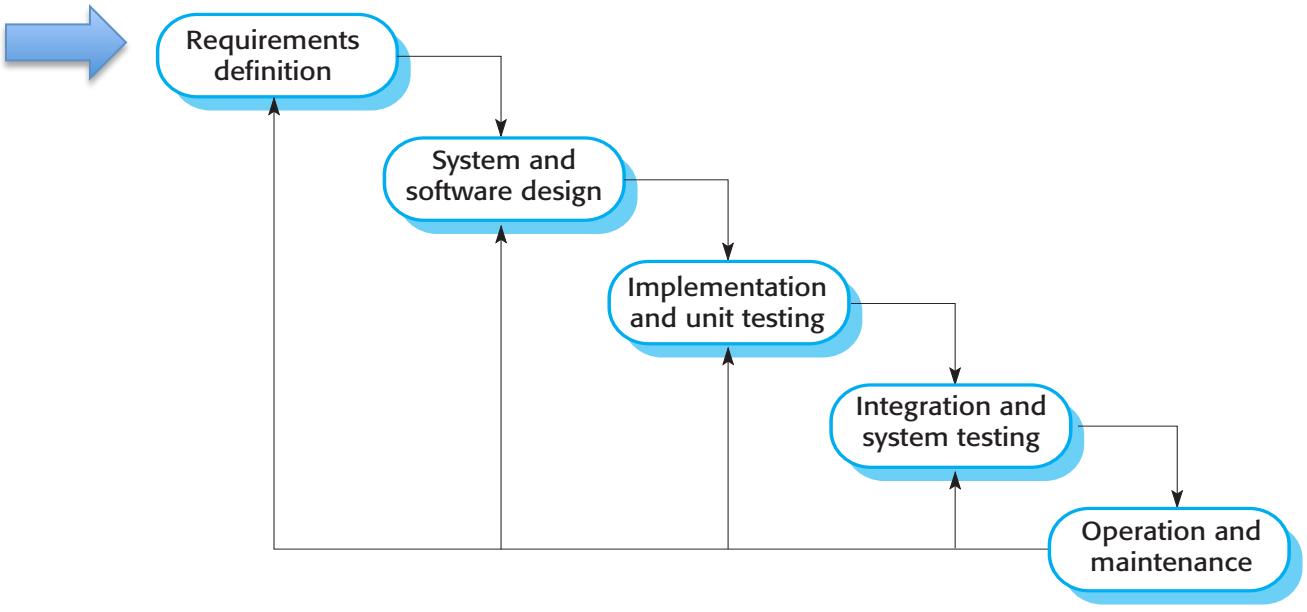
The Problem

There are numerous types of voting algorithms and in the United States, we typically use plurality voting where each voter is allowed to vote for only one candidate, and the candidate who polls the most votes is elected. It is rare for an election to be tied but if that occurs, there is typically a runoff between the tied candidates. For example, there have been three cases in history where there was a tie in the Electoral College for a presidential election. The House of Representatives then decided who was president by voting. For small sized, local elections a run-off may occur or even a coin flip can decide the outcome in some cases. Much research has been performed on voting theory and some believe that the Hare quota or Droop quota (ranked choice voting) is a better method than plurality. These types of voting algorithms are part of a family of algorithms called the single transferable voting (STV) systems.

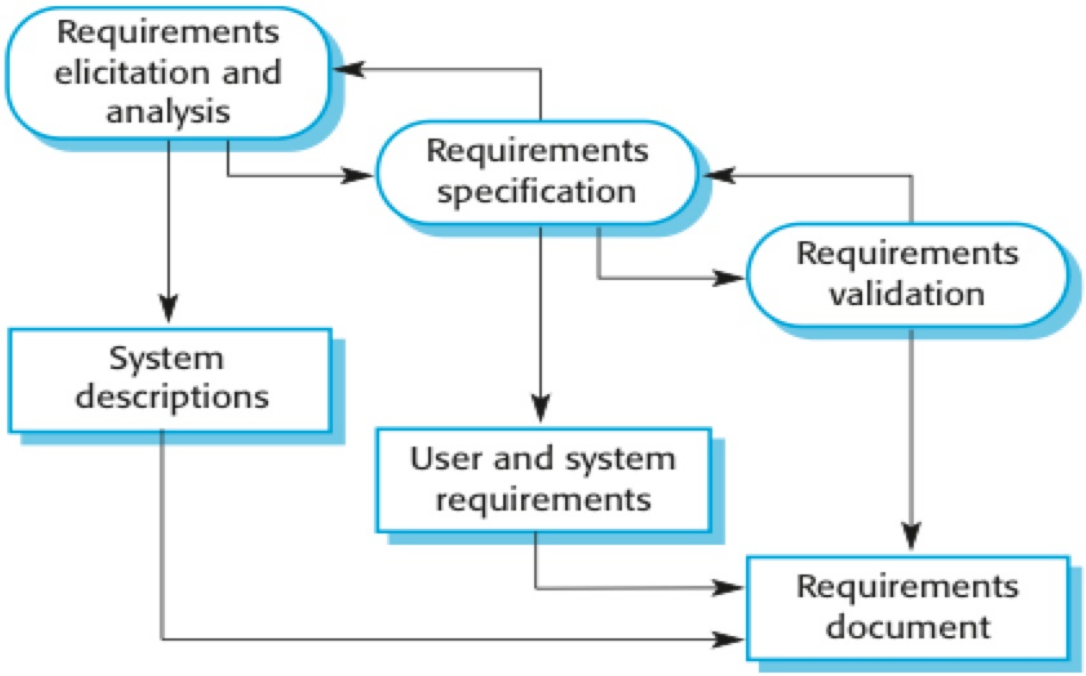
You are tasked with creating a voting system that is capable of performing both plurality voting and an STV system using the Droop quota. The program user will indicate what voting algorithm should be used (i.e. plurality versus STV.)

Your Work for This Piece of the Project

You and your team have been assigned to the task of developing this voting system and you will be using the Waterfall methodology.



Notice, that the first task is the requirements definition. You and your team will create the software requirements specification (SRS) document for this proposed voting system. The requirements engineering process will be used to guide your work.



Voting System Requirements

You are provided the following information from the users about the proposed system and what it needs to do:

- The actual voting will be done separately from the voting system you are developing. Ballots will be cast online and a comma delimited text file will be provided to you. You may assume that there are no numbering mistakes in the file (e.g. a voter will not accidently rank 1, 2, 5, 6, 7, 9 – notice the 3,4 and 8 are missing.)

- When the program is run, the user should be prompted to input how many candidates there are, how many seats there are to fill, the total number of ballots, and the algorithm to use (i.e. plurality or STV). You will need this information in order to calculate the Droop quota and to determine the type of voting algorithm to run (i.e. plurality versus STV.)

- Programs written in either C++ or Java will be accepted.

  - If you provide a C++ program, you will need to provide all source files and a make file. The program will be run from the command prompt.
  - If you provide a Java program, you must provide all source files and class files. Java programs will be run either at the command prompt or through Eclipse.
  - Your programs must run on a CSE lab machine.

- You will need to read in the file. How you choose to do this is up to you. Remember, it will be a comma separated values (CSV) file where each row is separated by a newline. The file will be exported from Excel into the CSV format. All preprocessing of the file will be done before you receive it. The first line of the file will provide you with the names of the candidates and each subsequent line will be a voter's ballot.

- For an STV ballot, each ballot must have at least one of the candidates ranked. For this iteration, you may assume that if there is a ballot in the file, it will have at least one of the candidates ranked. As you will see, the more rankings you complete as the voter, the more likely your vote will actually be counted towards electing the candidate(s) that you have as your top rankings. The long-term goal is for this system to be part of an integrated online voting system and all preprocessing will be done via the voting system itself that you are developing. You do not need to do any preprocessing of the file.

  Example File Generated for STV voting:

```
A,B,C,D,E,F
1,,2,,3,
3,2,1,4,6,5
1,2,,,3,4
,1,,,,
```

- For a plurality ballot, each ballot will only have one candidate indicated as choice for the vote. The ballot that will be generated will only have a single 1 in the candidate's position that was voted for by the voter.

  Example File Generated for Plurality voting:

```
A,B,C,D,E,F
1,,,,,
,,1,,,
,,,,,1
```

- The plurality algorithm should work as follows:
  - Each ballot will be assigned to a candidate and a count maintained for each candidate.
  - The candidate(s) with the greatest number of votes will be declared the winner(s).
  - If there is a tie for a given seat, you will randomly select the winner of that seat.

- The Droop Quota algorithm is as follows:
  - Step 1: Shuffle the ballots.
  - Step 2: Calculate the Droop Quota:

$$\left\lfloor \left(\frac{numberOfBallots}{numberOfSeats+1} \right) \right\rfloor + 1$$

  - Step 3: Distribute the ballots one-at-a-time into piles for the candidates. (The piles are ordered by the first vote received, not alphabetically.) If any candidate has reached the quota (for example, 22 ballots), they are immediately declared elected, those ballots are removed permanently from the process, and the elected candidate's name is recorded (in the order elected, not alphabetically). Subsequently, any ballot that follows in the distribution with their name listed goes to the next name on the list.
  - Step 4: After the first round of distributing the candidates, the candidate with the fewest number of votes is permanently eliminated and their ballots are redistributed. (In the case of a tie, the candidate who was last to receive their first ballot is eliminated.) As in the first round, if any candidate reaches quota they are immediately declared elected, etc. There may be multiple rounds repeating this process.

    - Step 5: You will continue this process until all ballots have been processed. You will have two lists, the elected and the non-elected. These lists will be ordered based on when they were elected and when they were placed on the non-elected list. It is possible for a "winner" of a seat to be on the non-elected list if they did not reach quota. The candidates placed on the non-elected list last will be higher up on the election list itself. You must provide both lists to the user.
    - Step 6: The user wants a report that acts as an audit for the election. This report should show the ballots that were assigned to a candidate as the election progressed. The report does not print to the screen but should be in a text file.

- The users want to be able to run a test file and the shuffle option should be able to be turned off. This will allow them to ensure the system is calibrated properly and also to allow for unit testing.

Your Project Task

Your project task is outlined above—develop specific sections of a requirements document for the SRSS and make use of use-cases to help you find the requirements. You will use the template provided on Moodle as a guideline to write up your SRSS. You are allowed to change/add/remove from the template but remember this template highlights the information you should include when developing a specification document. Your write up must be readable, changeable, and capture all the essential information we have discussed in class. We do expect to see use cases.

You have a requirements document template and use case template available on Moodle for the class.

Asking Questions

If you have questions about how the STV algorithm should work, you should document these questions in the section that needs this information in the document. You would use these questions to elicit more information during our two in-class interviews to be held on Monday, January 5th and Wednesday, January 7th (before the quiz).

Deliverables

You are required to turn in the requirements document and your use cases. You can put your use cases as a chapter in your requirements document, or you can submit it as a separate document.

Due Dates

SRSS requirements specification document and use cases due on: Sunday, February 11 at 11:55 p.m.