CSci 5801: Software Engineering I, Fall 2019
Project 1 – Waterfall Methodology
Software Requirements Specification (SRS) Document for Voting System
Due Date: Thursday, October 10th 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 included (e.g. Shana Watters, watt0087). Please use the name that is listed on the class roster so we will know who you are. You will upload your work to your team's GitHub repository (i.e. repo-TeamXX where XX is the team number) along with any supporting documents.

Special Instructions:
- Name this document **SRS_Team#.xxx** where the # is your team number and the XXX is the documents formatting extension (e.g. docx, pdf).
- If you have additional documents, name them using a similar naming convention (e.g. UseCases_Team1.pdf).
- Create a subdirectory under your repo-Team# called SRS. This will be where you put all documents generated for this assignment. Do not put your files on Canvas.

We will have an assignment location on to record your each team's scores and to store this project description. You will not upload your work to Canvas but put all documents for this assignment in your team repository under the SRS subdirectory.

We expect you to turn in your **typed** Software Requirements Specification document and your use cases that also must be typed. You can either incorporate your use cases in your SRS document or as a separate document. Please make clear in the document where to look for the use cases if they are not incorporated in the document itself. This portion of first portion of the Waterfall project is due at **11:55 p.m. on Thursday, October 10th**.

If you are not done, turn in what you have to GitHub by the due date and time. Anything received late will be given a 0. We can give you partial credit even if you are not fully finished.

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 count 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, and there are three types of voting systems: 1) Plurality/majority, 2) Proportional representative, and 3) Semi-proportional systems. We will be implementing 2 of these voting algorithms for this Waterfall project.

You are tasked with creating a voting system that is capable of performing two types of voting:

Party List Voting: Open Party List **AND** Closed Party List voting systems. The following overview is from FairVote.org and is taken directly from their website.

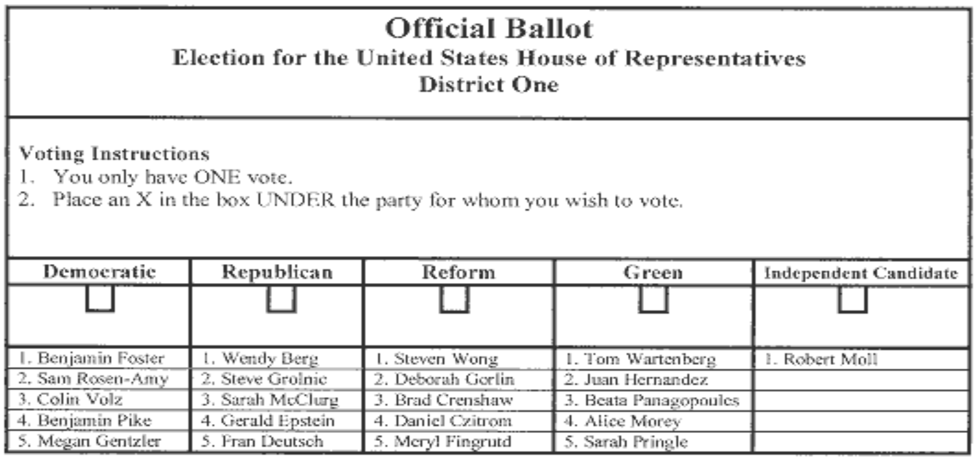
Party List Voting

Party list voting systems are by far the most common form of proportional representation. Over 80% of the PR systems used worldwide are some form of party list voting. It remains the system used in most European democracies and in many newly democratized countries, including South Africa.

How It Works. Legislators are elected in large, multi-member districts. Each party puts up a list or slate of candidates equal to the number of seats in the district. Independent candidates may also run, and they are listed separately on the ballot as if they were their own party (see below). On the ballot, voters indicate their preference for a particular party and the parties then receive seats in proportion to their share of the vote. So in a five-member district, if the Democrats win 40% of the vote, they would win two of the five seats. The two winning Democratic candidates would be chosen according to their position on the list.

There are two broad types of list systems: closed list and open list. In a closed list system--the original form of party list voting--the party fixes the order in which the candidates are listed and elected, and the voter simply casts a vote for the party as a whole. This is shown in the first ballot below, which illustrates an election for the House of Representatives in a five-seat district. Voters are not able to indicate their preference for any candidates on the list, but must accept the list in the order presented by the party. Winning candidates are selected in the exact order they appear on the original list. So in the example here, if the Democrats won two seats, the first two candidates on the pre-ordered list--Foster and Rosen-Amy--would be elected.

Closed Party List Ballot



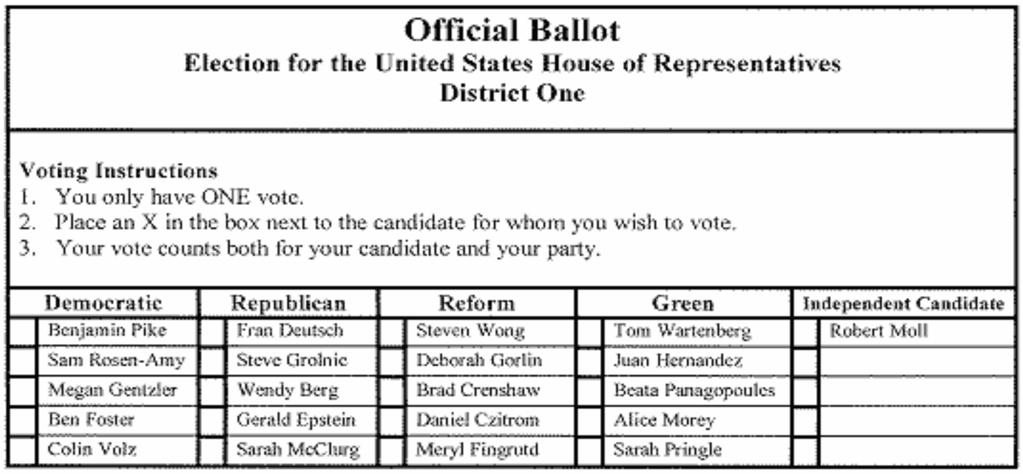
Official Ballot
Election for the United States House of Representatives
District One

Voting Instructions
1. You only have ONE vote.
2. Place an X in the box UNDER the party for whom you wish to vote.

Democratic	Republican	Reform	Green	Independent Candidate
☐	☐	☐	☐	☐
1. Benjamin Foster	1. Wendy Berg	1. Steven Wong	1. Tom Wartenberg	1. Robert Moll
2. Sam Rosen-Amy	2. Steve Grolnic	2. Deborah Gorlin	2. Juan Hernandez	
3. Colin Volz	3. Sarah McClurg	3. Brad Crenshaw	3. Beata Panagopoules	
4. Benjamin Pike	4. Gerald Epstein	4. Daniel Czitrom	4. Alice Morey	
5. Megan Gentzler	5. Fran Deutsch	5. Meryl Fingrutd	5. Sarah Pringle	

Most European democracies now use the open list form of party list voting. This approach allows voters to express a preference for particular candidates, not just parties. It is designed to give voters some say over the order of the list and thus which candidates get elected. One version of this is illustrated in the ballot below. Voters are presented with unordered or random lists of candidates chosen in party primaries. Voters cannot vote for a party directly, but must cast a vote for an individual candidate. This vote counts for the specific candidate as well as for the party. So the order of the final list completely depends on the number of votes won by each candidate on the list. The most popular candidates rise to the top of the list and have a better chance of being elected. In our example, if the Democrats won 2 seats, and Volz and Gentzler received the highest and next highest number of individual votes, they would rise to the top of the list and be elected. This example is similar to the system used in Finland and widely considered to be the most open version of list voting.

Open Party List Ballot



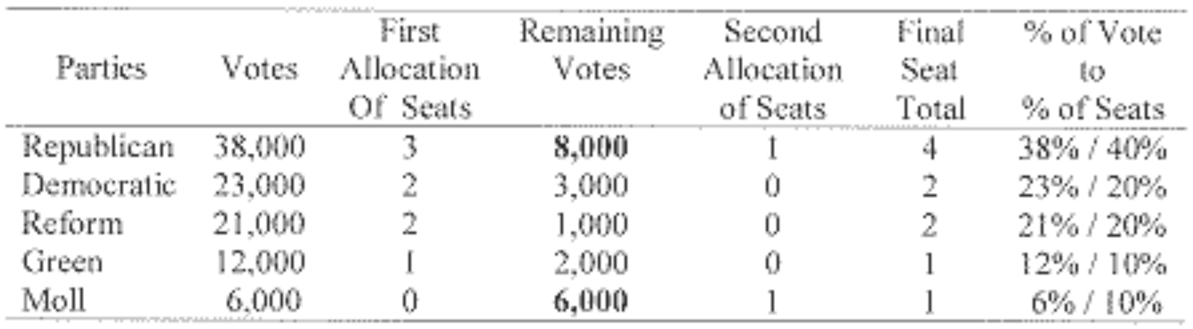
Official Ballot
Election for the United States House of Representatives
District One

Voting Instructions
1. You only have ONE vote.
2. Place an X in the box next to the candidate for whom you wish to vote.
3. Your vote counts both for your candidate and your party.

Democratic	Republican	Reform	Green	Independent Candidate
☐ Benjamin Pike	☐ Fran Deutsch	☐ Steven Wong	☐ Tom Wartenberg	☐ Robert Moll
☐ Sam Rosen-Amy	☐ Steve Grolnic	☐ Deborah Gorlin	☐ Juan Hernandez	☐
☐ Megan Gentzler	☐ Wendy Berg	☐ Brad Crenshaw	☐ Beata Panagopoules	☐
☐ Ben Foster	☐ Gerald Epstein	☐ Daniel Czitrom	☐ Alice Morey	☐
☐ Colin Volz	☐ Sarah McClurg	☐ Meryl Fingrutd	☐ Sarah Pringle	☐

A variety of different formulas exist for accomplishing the actual allocation of seats to the parties. One of the simplest seat allocation formulas is the called the "largest remainder formula." In this approach, the first step is to calculate a quota, which is determined by taking the total number of valid votes in the district and dividing this by the number of seats. In the example in the table below, 100,000 votes were cast and ten seats are to be filled. 100,000/10 = 10,000 – which is the quota. The quota is then divided into the vote that each party receives and the party wins one seat for each whole number produced. So the Republican party received 38,000 votes, which is divided by 10,000 to produce three seats – with a remainder of 8,000. After this first allocation of seats is complete than the remainder numbers for the parties are compared and the parties with the largest remainders are allocated the remaining seats. In our example, two seats remain to be allocated and the Republicans and Moll, the independent candidate, have the largest remainders, so they get the seats. Ultimately all the parties end up with the number of seats that as closely as possible approximates their percentage of the vote.

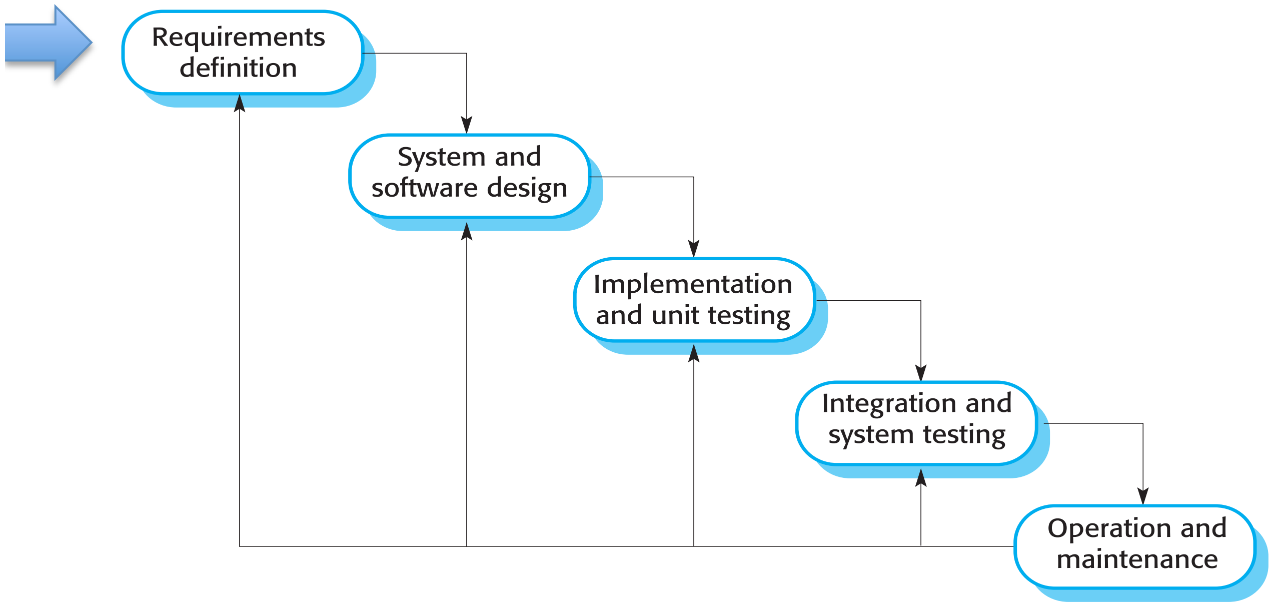
Largest Remainder Approach to Seat Allocation



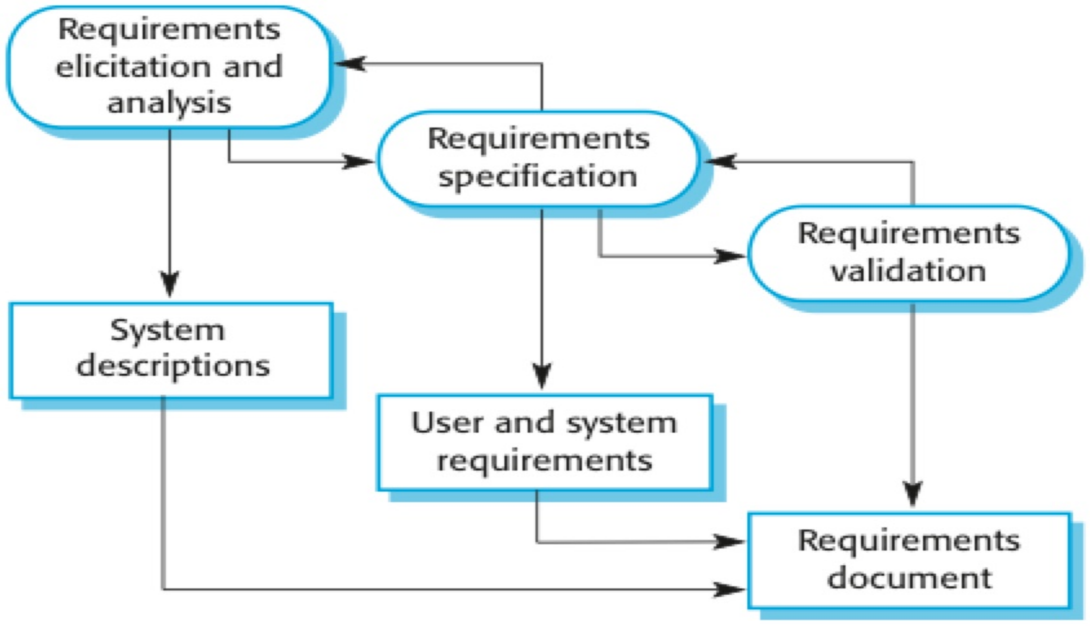
Parties	Votes	First Allocation Of Seats	Remaining Votes	Second Allocation of Seats	Final Seat Total	% of Vote to % of Seats
Republican	38,000	3	**8,000**	1	4	38% / 40%
Democratic	23,000	2	3,000	0	2	23% / 20%
Reform	21,000	2	1,000	0	2	21% / 20%
Green	12,000	1	2,000	0	1	12% / 10%
Moll	6,000	0	**6,000**	1	1	6% / 10%

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.


Requirements definition
System and software design
Implementation and unit testing
Integration and system testing
Operation and maintenance

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.


Requirements elicitation and analysis
Requirements specification
Requirements validation
System descriptions
User and system requirements
Requirements document

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 make any mistakes on the ballot.)

- You will only prompt the user for the filename. All information that is needed to run the program should come through the file or you determine based on the file. Assume the file will always be in the same directory as the program.

- When you prompt a user, you can have just text or a wonderful GUI--your choice.

- 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 comma separated values for the ballots where each row is separated by a newline. The file will be exported from Excel into the CSV format. The first line of the file will provide your team with the type of voting (i.e. CPL, OPL). See the formats below

- You will need to produce an audit file with the election information at the time (e.g. Type of Voting, Number of Candidates, Candidates, Number of Ballots, calculations, how many votes a candidate or party had, etc), you should list the winner(s), and you should show how the election progressed so that the audit could replicate the election itself. You should show who got what ballot and its order of being received if applicable.

- You should display to the screen the winner(s) and information about the election (e.g. type of election, number of seats). You do not need to show a long audit of the votes but do show the number of ballots cast, the winners and the stats for everyone or the party such as number of votes received.

- There will never be more than one file given to you per election.

- You need to write one program to handle both elections. Remember the election type is indicated on the first line of the file.

- For open party listing, all independents are grouped into one party.

- There will be no write in candidates for this system.

- You cannot change the file structure outside of the program since the election files will come in the predetermined format.

- If there is ever a tie, flip a coin. You must randomly select the winner in a fair coin toss.

- You will have programmers, testers, and election officials running and using this program. The results of the election could be shared with media personnel.

- You can assume we will use the most up-to-date CSELabs machines.

- Runtime constraints: An election should be able to run 100,000 ballots in under 5 minutes.
- The election file will be located in the same directory as the program.
- This program will be run multiple times during the year at normal election times and special elections.
- There are no special safety or security requirements. Security such as ensuring one vote for one person is handled at the voting centers.
- For an open party list ballet, voters express a preference for a particular candidate and a party. Each ballot will only have one candidate indicated as choice for the vote.

  Example File Generated for Open Party List voting:

```
OPL
3
9
6
[Pike,D]
[Foster,D]
[Deutsch,R]
[Borg,R]
[Jones,R]
[Smith,I]
1,,,,,
1,,,,,
,1,,,,
,,,,1,
,,,,,1
,,,1,,
,,,1,,
1,,,,,
,1,,,,
```

  1st Line: OPL for open party listing
  2th Line: Number of Seats
  3rd Line: Number of Ballots
  4th Line: Number of Candidates
  Following the 4th line are the candidates (one per line) and in the format above. The candidates and their party are in [ ]. Then the ballots are listed on per line and commas are used to separate the positions.

  You can assume no errors in the ballots. Only a single 1 will be placed in the position of the given candidate selected.

- For an open party list ballet, voters express a preference for a particular candidate and a party. Each ballot will only have one candidate indicated as choice for the vote.
- For a closed party list ballet, voters express a preference for a particular party. The order of the “elected” officials is dependent on the order of the party’s numbering of the candidates on the ballet.

Example File Generated for Closed Party List voting:

```
CPL
4
[D,R,G,I]
7
100
16
[Pike,D,1]
[Foster,D,2]
[Floyd,D,3]
[Jones,D,4]
[Mallory,D,5]
[Deutsch,R,1]
[Wong,R,2]
[Walters,R,3]
[Keller,R,4]
[Borg,R,5]
[Jones,G,1]
[Smith,G,2]
[Lewis,G,3]
[Smith,G,4]
[Li,G,5]
[Perez,I,1]
1,,,
1,,,
,1,,
,1,,
,,1,
,,,1
,,1,
1,,,
,1,,
,,1,
,,1,
,1,,
,1,,
and so on
```

1st Line: CPL for closed party listing
2nd Line: Number of Parties
3d Line: Parties in Order of Ballot Ordering
4th Line: Number of Seats
5th Line: Number of Ballots
6th Line: Number of Candidates
Following line 6 are the candidates' information followed immediately by all of the ballots. The order of the party is indicates their position in the ballot.

You can assume no errors in the ballots. Only a single 1 will be placed in the position of the given party selected.

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 Canvas 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 Canvas for the class.

Asking Questions
If you have questions, you should document these questions in the section that needs this information in the document. You will use these questions to elicit more information during our in-class interviews to be held on Tuesday, October 8th. In addition, I will post answers to questions on Tuesday, October 1st. I will be away at a conference but will answer any questions posted to the discussion board by 6:00pm, Tuesday, October 1st. I will only answer those questions posted by that time since you will then have a follow up in-person interview then next week. I have tried to anticipate common questions and have included details in this document.

Deliverables
You are required to turn in the requirements document that you created using the IEEE Software Requirements Specification Document Template and your use cases that you created using the Use Case Template. You can incorporate your use cases in your requirements document, or you can submit them in a separate document. You must add all of your documents to

github.umn.edu/umn-csci-5801-f19/repo-Team#/SRS We will only look in your team repository for your work.

Due Dates

SRSS requirements specification document and use cases due on: Thursday, October 10 at 11:55 p.m.