**Rochak Kunwar**

**Effective Size Estimation**

1. The code is itself not very complicated, and has high readability and manageability. It is just around 150 lines of codes, but there is lot more to be added to achieve our target. Specially a lot of logic needs to be gone in did win function(may be around 20 lines of codes), and one big loop in main, and some #define macros and cin - cout. That should be enough with some changes in the code. So here is my estimation for

\*\*\*Let’s assume that 1 SLOC is equivalent to 1 logic \*\*\*

* 1. Snew = 20 + 3 + 5 + 5 = 33 SLOC
  2. Smod = (150 – 20) = 130 (The small changes specially board size and X, 0, .)

Around 40 lines of codes are useless they are just space or comments. So, Smod = 130 – 40 = 90 SLOC

* 1. Sreused = There are 10 lines of codes. Out of those 100 are only the actual code. From there 40 SLOC will be changed which leaves us with 60 lines of real code. So , I would say may be we will have 60 SLOC as Sreused.

Smin, Sml , Smax , Smean ????

1. For Snew : As I already said it will come from logics in did win function, ,macros, cin to ask user for the input, loops in main.
   1. Smin: 10 lines in did win, 3 lines in loop, 3 lines in getting data from user and assigning to variable, and 4 lines for const. So Smin= 10 + 3 + 3 + 4 = 20 SLOC
   2. Sml : 25 SLOC
   3. Smax: 40 SLOC
   4. Smean : (20 + 4 \* 25 + 40)/ 6 = 27
2. For Smod: There is a little confusion on this one. According to the reading it is said that Smod is total lines – lines changes.
   * 1. Smin: According to the formula higher the change lower the Smod because Smod = total codes – change. So, if we change minimum of 15 logics then our Smin will be 150 – 15 – 40(comments and space) = 95 SLOC.
     2. Sml : I feel like we will do lot of changes in the code even though its not needed. And I think those lines should also be counted. In any project we change like atleast 5% of the code that we don’t even need to. So, I think I will change (15 necessary change + 10 unnecessary change). Which will make my Sml to 150 – 25(necessary + unnecessary) – 40(comments and space) = 85 SLOC.
     3. Smax: I feel like if we don’t understand the error then we end up doing lots of unnecessary modification. So I aim high for this; around 50 SLOC. So Smax = 150 – 50 – 40 = 60 SLOC.
     4. Smean = (95 + 4 \* 85 + 60 ) / 6 = 83
3. Sreused:
   1. Smin: If we end up making lots of unnecessary changes chances is reusability will drop down so, I will keep it as 150 – maximum changes in Smod – 40 – some deleted code = 150 – 50 – 40 - 5 = 55 SLOC

After adjusting asd assuming we did a lot of unnecessary modification : Smin : 40 SLOC

* 1. Sml : Just by estimating 13 lines for each function except did\_win it will be 52 SLOC.
  2. Smax : Adding some stuff outside function. Overall it can be maximum of 150(total lines) – 40(space and comments) – 15(minimum changes in code) – deleted code = 90 SLOC
  3. Smean = (40 + 4\* 52 + 90)/6 = 56

1. A. SI1
   * 1. Fdes: 0.5
     2. Fimp: 0.1
     3. Ftest: There is not much we can reuse without modifying : 0.05
   1. SI2:
      1. Fdes: 1.0
      2. Fimp: 0.05
      3. Ftest: There is nothing except function definition so what to test?? 0.05
   2. SI3:
      1. Fdes: 0.2
      2. Fimp: 0.05 (Everything is given put in the loop)
      3. Ftest: We reuse the whole code inside a main. I want to test them them all except writing and reading so 0.5
2. From the reading we get a equation called “The simplified Jensen effective size” which has a constant in its equaltion. We will use the same equation.

SAF1 = 0.4Fdes + 0.25Fimp + 0.35Ftest = 0.4 \* 0.5 + 0.25 \* 0.1 + 0.35 \* 0.05 = 0.2425

SAF2 = 0.4 \* 1.0 + 0.25 \* 0.05 + 0.35 \* 0.05 = 0.11

SAF3 = 0.4 \* 0.2 + 0.25 \* 0.05 + 0.35 \* 0.5 = 0.27

1. Seffective :

Seffective = Snew + Smod + Sreused SAF

Since I have done combined Snew, Smod, and Sreused in step 1 I will take an average of SAF for this problem.

SAF = ( 0.2425 + 0.11 + 0.27 )/ 3 = 0.21

Seffective = 27 + 83 + 56\* 0.21 = 122

1. T = 121/ 50 = 2.5 hours

**Experiment:**

**Design :** It took me 40 mins to analyze the code and come up with the pattern to do the task. I realized that we were not supposed to make this game a playable. It was just comparison of a board to check whether X wins or Y wins. In design phase I realized the implementation should not take more than 30 mins.

Implementtation: The implementation was fairly simple. I never used #define because I haven’t used that before. I used few const to make a size of the board and realized that I need to make my own board for different size. Then I asked user for all the entry he wants in the board and checked his own entry to test whether X or O wins. I didn’t make any changes in X and Y I left it as hardcoded. But I changed the value 3 from all the program to some variable that user wants which took maximum time. The did win function I copied from stack Overflow and did very little change to make the whole program work.

I added around 60 lines of new code

I changed around 16 lines of codes only. Mostly my changes were size of board.

I reused 150 – 40 – 16 = around 90 lines of codes and didn’t delete anything.

Testing: Since the program was very small and all the code was done in emacs. I didn’t have tools to do advance testing. I simply did cout in few places and changed my code accordingly. The test was more like a unit testing where I was checking the result of different functions and logics.

Conclusion:

The estimation was way off. Because I was thinking that I have to make a program run until someone wins. I visualized it as game more than just testing. Which made my estimation little off. Plus my did\_win function came from stack overflow which also affected my result. Overall it tool me 1:30 hours to figure out the whole solution including googling and designing. Which was only 1 hour less than estimation. I feel like the result would have ben very close if I have done all the logics and coding.

The values below can be up and down by few digits

My Seffective = Snew + Smod + Sreused SAF

Seffective = 60 + 16 + 90\* 0.21 = 94.9

T = 94.9 / 50 = roughly 2 hours

I am confident this time would have been exact time if I knew the logic of did\_win function and just had to code and test.

Difference in Seffective = 122 – 95 = 27

Difference in Time = half an hour 0 if adjusted(I mean if I have not copy pasted)

efficiency:

There is a problem in this. And the problem is size of project. It is very small to do individual. If you do SI2 and SI1 you pretty much know what needs to be done in SI3. So. I am doing this as a combined.

It took me 40 minutes to write the whole code and I added 60 new lines and changed around 16 lines. Since copy pasted around 40 lines of code, but it took me some time to figure out the logic so I will adjust 60 lines of codes to 15 lines which makes my total lines of code modified and added to : 16 + 20 = around 36 lines.

Time taken for one line of code = 36 lines / 40 minutes = 0.9 around 1 line/SLOC per minute.