Ponder 09 : Size Estimation

Jacob Thomas and Ryan Dockstader

For all **estimations**, we are using the following calculations for Snew Smod and Sreused :

* Main – 25 Lines
* Read – 21 Lines
* Write – 17 Lines
* Display – 28 Lines
* DidWin – 1 Line
* Global – 8 lines

(25 + 21 + 17 + 28 + 1 + 8)/5 = 16.6666

Since we can’t have half a line of code, we rounded up to give an average SLOC per module (function in this case) of **17 lines.**

# S1: Macros

## Step 1: S variables

### SNew

We will not be writing any new functions, only making modifications to existing functions. So Our Snew = 0 SLOC.

### Smod

We will need to modify the following functions

* Read
* Display
* Global

So, Smod = 3 \* 17 = 51 SLOC

### SReused

The remaining functions that will not be modified are:

* Main
* Write
* DidWin

So, Sreused = 3 \* 17 = 51 SLOC

## Step 2: F variables

### Fdes

0 out of 3 functions will need to be reverse engineered, since none of them are directly interacting with the functions we will be making modifications to.

Fdes = 0/3 = 0

### Fimp

There is a small amount of interaction between the functions that we will be modifying and the existing functions, so the interface effort is mostly taken care of but we will need to verify the Main function is still interfacing the functions correctly after the modifications are made.

Fimp = .02

The ratio between the interface effort and the effort required to produce the interface definitions from scratch : 0.2Fimp

### Ftest

We don’t have access to any of the testing documentation, so Ftest = 1.0 - 0 = 1

## Step 3: SAF

SAF = 0.4(0) + .25(.02) + .35(1) = 0.355

## Step 4: Seffective

Seffective = 0 + 51 + 51(0.355) = 67.105

Seffective = 67 (rounded up)

## Step 5: T

T = 67 / 50 = 1.34 hours

# S2: Winning

## Step 1: S variables

### SNew

We will not be writing any new functions, only making modifications to existing functions. So Our Snew = 0 SLOC.

### Smod

We will need to modify the following functions

* DidWin

So, Smod = 1 \* 17 = 17 SLOC

### SReused

The remaining functions that will not be modified are:

* Main
* Read
* Write
* Display
* Global

So, Sreused = 5 \* 17 = 85 SLOC

## Step 2: F variables

### Fdes

1out of 5 functions will need to be reverse engineered, since none of them are directly interacting with the functions we will be making modifications to.

Fdes = 1/5 = 0.2

### Fimp

There is interaction between the functions that we will be modifying and the existing functions, so the interface effort is mostly taken care of, but we will need to verify the Main function is still interfacing the functions correctly after the modifications are made.

Fimp = .06

### Ftest

We don’t have access to any of the testing documentation, so Ftest = 1.0 - 0 = 1

## Step 3: SAF

SAF = 0.4(0.2) + .25(.06) + .35(1) = 0.445

## Step 4: Seffective

Seffective = 0 + 17 + 85(0.445) = 54.825

Seffective = 55 (rounded up)

## Step 5: T

T = 55 / 50 = 1.1 hours

# S3: Three Dimensions

## Step 1: S variables

### SNew

We will not be writing any new functions, only making modifications to existing functions. So Our Snew = 0 SLOC.

### Smod

We will need to modify the following functions

* Read
* Write
* didWin
* Display
* Main
* Global

So, Smod = 6\* 17 = 102 SLOC

### SReused

There are no reused functions

So, Sreused = 0 SLOC

## Step 2: F variables

### Fdes

There are no reused functions, but every one of them will need to be redesigned to work with 3 boards instead of 1

Fdes = 1

### Fimp

There is a large amount of interaction between all the code that will be modified.

Fimp = .1

### Ftest

We don’t have access to any of the testing documentation, so Ftest = 1.0 - 0 = 1

## Step 3: SAF

SAF = 0.4(1) + .25(.1) + .35(1) = 0.775

## Step 4: Seffective

Seffective = 0 + 102 + 0(0.775) = 102

Seffective = 102

## Step 5: T

T = 102 / 50 = 2.04 hours

# Conclusion Set

## SI 1 : Macros

Estimations

* Estimated Completion Time: 80 Minutes
* Total Effective Size of change: 67 SLOC

Actual

* Actual Completion Time (16 minutes)
  + Design Phase: 8 minutes
    - Researched the #define and how it worked. Determined that it was a direct replace macro, with the syntax being #define MACRO\_NAME REPLACE\_TEXT
    - There wasn’t a ton of resources online to answer direct questions so most of this time was just researching in the cpp reference.
  + Implementation Time: 4 minutes
    - Minimal lines had to be added/changed. There were 4 #define lines added, and then several X’s, O’s, and periods. The biggest modification was the board\_size variable.
  + Testing Time- 4 minutes
    - Used all the provided test cases, excluding the ones build for a 3d board.
* Actual Total Effective Size of change: 25 SLOC
  + The difference here was that a lot of the lines didn’t necessarily have to be modified, since it was a direct replace (magic strings w/ the macro). This reduced the change, and make it significantly faster to perform the change

## SI 2 : DidWin implementation

Estimations

* Estimated Completion time: 66 minutes
* Total Effective Size of Change: 55 SLOC

Actual

* Actual Completion Time (23 minutes)
  + Design Phase (5 minutes)
    - This consisted of determining the win conditions and deciding if it was worth the extra effort for the single loop performance benefit. It was determined that for the data sets we’re working with; it wouldn’t be worth it. So, the decision was made to make multiple loops
  + Implementation Phase (13 minutes)
    - This was all the modifications to the Did Win function. Since there were more loops to build, it took longer. But not as long as formulating all the logic into the single loop would have taken
  + Testing Phase (5 minutes)
    - This consisted of testing all the provided test cases, other then the 3d ones. I did run into a bit of a snag on the diagonal, but this was resolved by moving my counter outside of the loop.
* Actual Total Effective Size of Change: 45 SLOC
  + This was pretty close to the estimated change. I think had I added brackets to every if and for, we would have been nearly right on with the estimation.

## SI 3 : 3D

Estimations

* Estimated Completion Time: 124 Minutes
* Total Effective Size of change: 102 SLOC

Actual

* Actual Completion Time (120 + minutes)
  + Design Phase: 60 minutes
    - Read through and looked at where the first implication of the 3d board would need to go. Determined there would need to be an if statement to let the program handle two different kinds of arrays.
    - Calculated how many statements would be needed to compare every board against the first board and each possibility in 3D space. There were tons.
    - Psuedocoded out the steps I felt were going to be needed on paper.
    - Got an opinion about my current ideas from an industry professional.
  + Implementation Time: 60+ minutes
    - This took me a lot longer than I expected it to. I had great difficulty finding where I should start and knowing how to modify it.
    - Got stuck when realizing how complex the change would need to be to account for both types of boards.
    - Implemented a change throughout every function to modify the board to be 3D.
    - Modified each sub function to take in the 3D array
    - Ran and found many bugs
    - **Determined that I did not have the time to continue to work on the code and the project would be unfinished.**
  + Testing Time- 0 minutes
    - The testing was not performed due to a lack of time. The project will not run 3D boards.
* Actual Total Effective Size of change: 29 SLOC
  + Everywhere that had reference to the 2 current arrays needed to be modified to take the 3rd.
  + Every function call that passed the two arrays as parameteres needed to be modified to take the 3rd.
  + Every for loop had to be placed inside another for loop.
  + Instantiation of the board was modified to take another dimension.
  + NOT COMPLETED: the many many conditional statements required to filter down correct 3D board guesses.
  + NOT COMPLETED: the global conditional statements that would have allowed for both types of boards to be submitted.

In overall conclusion, the calculations were wildly inaccurate for our team. SI 1 and 2 had an estimation time double what we actually saw, and SI 3 estimation of 2 hours was not nearly enough. It was most accurate for SI 2, and we believe this to be because the SI itself was so small. We may conclude than, that the smaller the SI estimated, the more accurate the estimation is likely to be. This is however obvious, as smaller numbers have smaller percentage differences.