Rochak Kunwar

Anthony Wagner

Josué Ruiz

## PONDER 08 : ESTIMATE COST

Every software engineer will be asked to make estimates as to how long it will take to complete a given task. Sometimes these estimates are rough and off-the-cuff and are used by managers and designers to weigh design decisions. Sometimes these estimates need to be precise and accurate — used for resource allocation and budgeting purposes. The reading this week discussed a methodology for creating this second type of estimation. Your assignment this week is to create a reliable estimation of a real-world software project.

### Integrated Graduation Planner

Often software engineers are asked to give detailed and accurate estimations based on very terse project descriptions. You are asked to do this week. Consider the following project description:

Create a system to integrate the graduation planner with the registration system so that when you complete the graduation plan, it will register you for your classes. Since everyone has their graduation plan done early, it will enable forecasting for the departments (for example, the number of sections of a class to offer in a given semester). When your day it comes to register, you are automatically registered for the class unless you desire to change your schedule.

A detailed description of the requirements are available here: [Registration and Graduation Advisor for BYU-Idaho](https://content.byui.edu/file/1682c7bf-acb5-4616-aeae-34930a15bee6/1/Ponder/416.08.Ponder.SRS.pdf). You may feel that this project description is not sufficient. This is true! You may make clarifying questions to the stakeholder (the instructor in this case). He or she may answer your question or may not! (this is a real-world possibility!). Regardless of the quality of the answer, you still need to make a high-quality estimation.

Please take the following into consideration:

* There will be ten software engineers working on the project.
* Each of these engineers is "experienced," meaning they have been working on similar projects for five years and have ten years of industry experience. The team collectively has experience with system, web technology, server technology, and databases.
* The average annual salary for each of the ten will be $100,000.00 a year.

### Two Levels of Estimation

You need to make a system-level estimation of the project and a component-level estimation of one of the SIs. Of course, the final number you come up with is not as important as the methodology you follow to arrive at the number.

### System-Level Estimation

Create a system-level estimation of the entire system. To make sure that everyone is on the same page, there will be five SIs in this project. Please note that the SRS describes potentially more than five SIs. In other words, each of the above can be subdivided into smaller SIs. Generally speaking, the smaller the SI the more accurate the estimate. Please do not subdivide these big SIs into smaller ones. That is more work than necessary for this project. Our purpose is to understand the process, not come up with an accurate estimate. The five SIs are the following:

* Course Choice Display (3.2.1)
* Filtering (3.2.2)
* Registration Restrictions (3.2.3)
* Scheduling Conflicts (3.2.4)
* Waiting List (3.2.5)

Use the methodology described in the reading. Justify all your decisions and explain where all the numbers originated. When finished, you must have an estimate for how long it will take to complete the project.

Follow the steps outlined on page 28 of the reading. A couple hints:

1. The five SIs are given.

* Course Choice Display
* Filtering
* Registration Restrictions
* Scheduling Conflicts
* Waiting List

2. See Table 4-2 or Table 4-5 for the complexity value. It could be different for each SI.

* Course Choice Display (Will display all the course semester wise the student have selected for particular semester
  + **Complexity Value: 21**
* Filtering: Having only selected data show up in the table. Teachers, Name of class, ect.
  + **Complexity Value: 28**
* Registration Restrictions: Some students might have hold on their account. Or they may signup up for unnecessary classes, some do not want to graduate because they don’t have a job and will take unnecessary classes so they won't get kicked out of school. All this condition needs to be studied in detail.
  + **Complexity Value: 21**
* Scheduling Conflicts: One student cannot sign up for two classes at the same time. One cannot retake classes more than certain time. No more than two religion classes in one semester. Changing of major also need to be considered. Changing of track(semester), etc.
  + **Complexity Value: 28**
* Waiting List: There are always a lot of students every semester in a waiting list how to deal with such situations. A system need to be trained on that.
  + **Complexity Value: 28**

3. You will need to multiply the number of procedures described in the SRS by 2,000 (in section 6.4.2.3 of the text), which we will read about next week. Don't forget to take into account the growth factor (Table 4-4).

* S (Course Choice Display) = 9 SLOC \* 2000 = **18000 ESLOC**
* S (Filtering) = 7 SLOC \* 2000 = **14000 ESLOC**
* S (Registration Restrictions) = 4 SLOC \* 2000 = **8000 ESLOC**
* S (Scheduling Conflicts) = 4 SLOC \* 2000 = **8000 ESLOC**
* S (Waiting List) = 8 SLOC \* 2000 = **16000 ESLOC**

4. The productivity factor *Ck* needs to be in hours per ESLOC. The table in 4.4.1. is in ESLOC / month where there are 166 hours per month (2,000 hours per year / 12 months).

(We assumed that one developer can do 100 lines of effective code in a day/ 500 lines a week and 2000 lines in a month. Regarding the ESLOC we found out that much of the code already exist and many ESLOC could be reused. This means that the actual ESLOC will be lesser than what we calculated.)

* Ck(Course Choice Display): 18000/8.8 PM = **2045 per person Month**
* Ck(Filtering): 14000 / 7 PM = **2000 per person month**
* Ck(Registration Restrictions): 8000 / 4.4 PM = **1818 per person Month**
* Ck(Scheduling Conflicts): 8000/4.4 PM = **1818 per person Month**
* Ck(Waiting List): 16000/ 8.8 PM = **1818 per person Month**

5. Compute the *EDevelop*, estimate the *k*, then compute the *ETotal* for the project (see Section 4.7). When finished, compute the cost is in dollars.

* Course Choice Display : **k = 1.35**
* Filtering : **k = 1.28**
* Registration Restrictions : **k = 1.34**
* Scheduling Conflicts : **k = 1.30**
* Waiting List: **k = 1.29**

Estimation of development effort Person Month(We assumed that all other features are already developed like, database of classes in particular classes, and the below listed are additional features. Also we assumed that one developer can do 100 lines of effective code in a day/ 500 lines a week and 2000 lines in a month)

* Course Choice Display : **Edevelop = 18 PM**
* Filtering : **Edevelop = 14 PM**
* Registration Restrictions : **Edevelop** **= 8 PM**
* Scheduling Conflicts : **Edevelop = 8 PM**
* Waiting List: **Edevelop = 16 PM**

Etotal = Summation ( Etotal for each SI)

= (1.35 \* 18) + 1.28\* 14 + 1.34 \* 8 + 1.30 \* 8 + 1.29\* 16

= **83.98 PM => 83.98 /12 \* 100,000 = $700, 000**.

6. After you have followed the methodology of the reading, make any necessary modifications based on what you have learned from the class debate. Give a brief rationale as to why the adjustment was necessary and why you feel the new estimation is more accurate. This part is not optional; you must suggest an alteration. For example, the text suggests that a developer can write about 131 lines of code per month. Since a month has about 166 hours, that is about one line of code per hour. Might this be a bit pessimistic? When finished, you should have a refined estimate measured in months. This is a very important part of the assignment, constituting a full 20% of the grade.

I knew a regular developer does far less coding than what we do in school. But we estimated that a developer can do 1000 lines of code in 1 month. Since the actual fact is average developer takes much more time than what we have estimated. It is very important for us to modify our estimate.

Our calculated result is **83.98 PM**.

So, the ratio estimation of coding by us and by actual developer is  
 2000/ 135 = 14.8 times of actual developer

So, Our adjusted development effort is 14.8\*83.98 = 1244 PM (Which seems pretty high for just a simple modification in software. My concern is with the lines of code for each requirement. I feel like that is too much for just some modification in software. So, we are going with our values than what is suggested in the book. May be developer will write 135 lines of code by himself in a month, he can reuse total of 1000-2000 lines in a month. So, we will go with our estimation.)

Also, we feel like usually projects take longer than estimated or there might be certain situations where key developers might get sick or leaves the company so we have increased our development effort to reflect this situations resulting in 18 PM. So, the final **Ed = 90 PM**.

If each developer is paid $100,000 for a year, then the total budget for the project results in **100,000/ 12 \* 90 = $750, 000**.

7. You can skip step 7; we will not do the staffing equations or compute *Td*.

### Component-Level Estimation

Create a component-level estimation of the Filtering (3.2.2) SI component of the integrated graduation planner project. As with the system-level estimation completed earlier, you will need to both follow the methodology described in the text and create a refined estimation based on improvements to the methodology. Again, all the numbers, processes, and improvements must be fully justified.

You may have some difficulty coming up with specific values for many of these variables. Most of these are described in more detail in later chapters of the reading. At this point in time, you do not need to master these later chapters. Please just come up with a coarse estimate for the values.

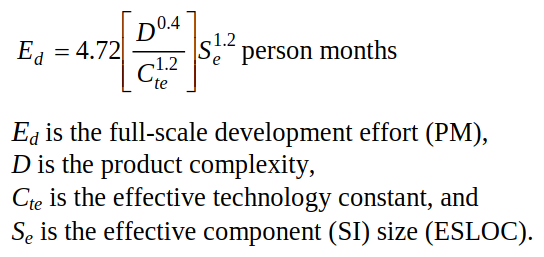
Follow the steps outlined on page 40 – 41 of the reading. A couple hints:

1. Though the reading asks us to come up with a list of the elements in each SI (corresponding to the functions listed in the SRS), you will only need to work at the SI level. Estimation at the element level will be much more accurate but take more time than we have.
   1. Course Choice Display : week PM
   2. Filtering
   3. Registration Restrictions
   4. Scheduling Conflicts
   5. Waiting List
2. Use the complexity number from the System-Level estimation.
   1. Course Choice Display : 21
   2. Filtering : 28
   3. Registration Restrictions: 21
   4. Scheduling Conflicts: 28
   5. Waiting List: 28



* 1. Course Choice Display : week PM Td = X:21 \* squareRoot( Ed:8.8PM) months = **62.3 months**
  2. Filtering : Td = X:28 \* squareRoot( Ed:7PM) months = **74.08 months**
  3. Registration Restrictions Td = X:21 \* squareRoot( Ed:4.4PM) months = **44.04 months**
  4. Scheduling Conflicts : Td = X:28 \* squareRoot( Ed:4.4PM) months = **58.73 months**
  5. Waiting List : Td = X:28 \* squareRoot( Ed:8.8PM) months = **83.06 months**

1. The effective size estimation for the Component-Level will be the *Se* numbers from the System-Level estimation.
   1. S (Course Choice Display) = 9 SLOC \* 2000 = **18000 ESLOC**
   2. S (Filtering) = 7 SLOC \* 2000 = **14000 ESLOC**
   3. S (Registration Restrictions) = 4 SLOC \* 2000 = **8000 ESLOC**
   4. S (Scheduling Conflicts) = 4 SLOC \* 2000 = **8000 ESLOC**
   5. S (Waiting List) = 8 SLOC \* 2000 = **16000 ESLOC**
2. The development environment is encapsulated in *Cbt*, the effective technology constant. Though we will spend a full week discussion this, you can get a rough estimate from Figure 8-1 and 8-2.
   1. ***Cbt*= 8000**
3. The development effort section is 5.5 in the reading.



1. Course Choice Display : week PM Ed = 4.72 \* [(D:21 \* 0.4)/Cte: 8000 \* 1.2)]\* Se 18000 ESLOC : person months = **74.34 person months**
2. Filtering : Ed = 4.72 \* [(D:28 \* 0.4)/Cte: 8000 \* 1.2)]\* Se 14000 ESLOC : person months = **77.09 person months**
3. Registration Restrictions : Ed = 4.72 \* [(D:21 \* 0.4)/Cte: 8000 \* 1.2)]\* Se 8000 ESLOC : person months = **33.04 person months**
4. Scheduling Conflicts : Ed = 4.72 \* [(D:28 \* 0.4)/Cte: 8000 \* 1.2)]\* Se 8000 ESLOC : person months = **44.05 person months**
5. Waiting List : Ed = 4.72 \* [(D:28 \* 0.4)/Cte: 8000 \* 1.2)]\* Se 16000 ESLOC : person months = **88.10 person months**
6. The development schedule will be your *Td* value.
   1. Course Choice Display: **62.3 months**
   2. Filtering: **74.08 months**
   3. Registration Restrictions: **44.04 months**
   4. Scheduling Conflicts: **58.73 months**
   5. Waiting List: **83.06 months**
7. You can skip step 7.
8. As with the System-Level estimation, you will find certain irregularities and flaws in the numbers and equations of the text. Please validate the computed values against your intuition and experience. Make any necessary corrections and justify them. This is a very important part of the assignment, constituting 20% of the grade.

**Total Td** = **Around 321 MP**

The estimation seems accurate according to the component level estimation. Usually the project initial estimation is less than what we anticipated. Even big companies take double the time they allocate for software development. This happens especially at the beginning where they say they can finish the tasks in a year but they end up taking 3 years. Some examples of those incidents are SPACEX first flight, tesla Roadster, and we can find many examples like those in the industry. As the time increases, so does the cost of development. Our initial estimation was **$750,000** but we came up with this much higher figure 321MP \* $100,000 /12 = **$2,675,000**. We still believe, even though our cost is higher, this project is doable inside **1 million dollar.**