CS 430 - Lecture 21 - Planning and Estimation I - Function Points

Collin Roberts

November 28, 2023

Outline

When To Estimate

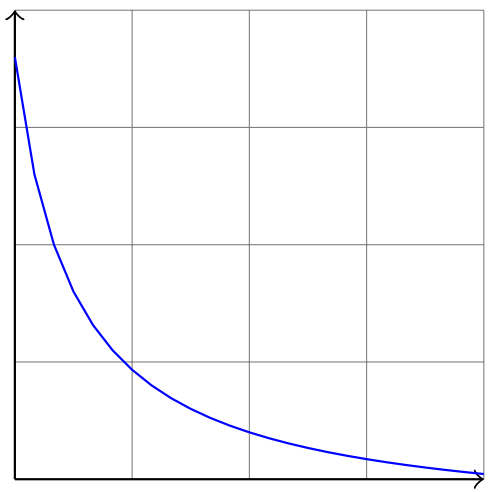
after requirements workflow - only an informal understanding of what is needed

1. At this point, our ranges of estimates must be broad.
2. Figures 9.1 & 9.2 explain somewhat why this is true.

When To Estimate

3. This is a summarized Figure 9.1 from the text. It displays a model for estimating the relative range of a cost estimate for each workflow.

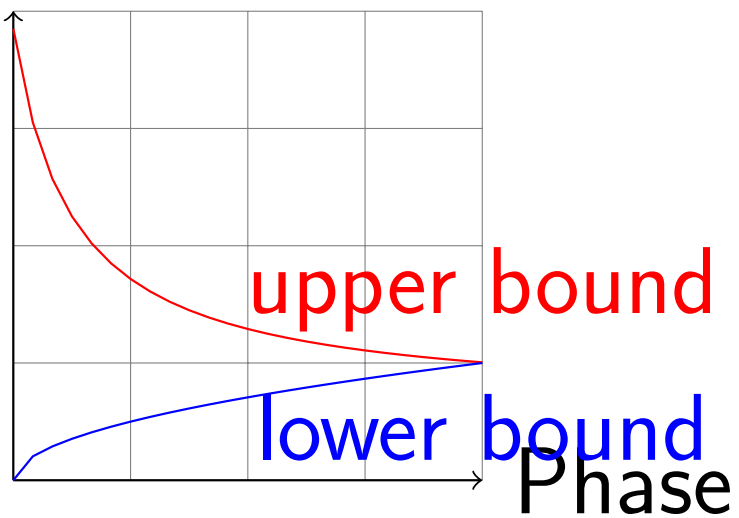
Relative Range of Cost Estimate



Phase

When To Estimate

4. This is a summarized Figure 9.2 from the text. It displays the range of cost estimates, in millions of dollars, for a software product that costs $1 million to build.



When To Estimate

5. We provide a preliminary estimate here, so that the client can decide whether to proceed to analysis or not.

When To Estimate

after analysis workflow - a more detailed understanding of what is needed

1. For the rest of Ch 9, we assume that we are estimating at this point.

Remarks:

1. In practice, you may find yourself getting pressured by the client to reduce your preliminary estimates, to ensure that the project goes ahead. Common sense says that a client cannot dictate both the requirements and the costs to satisfy them. If the client thinks that the preliminary estimates are too high, then they can:
    1. reduce the scope of the requirements, to reduce the estimated cost, or
    2. increase the total budget.

    Giving in to pressure to reduce estimates at this point ALWAYS leads to problems later on.

Estimating Cost

All Costs of Development:

1. **internal**, i.e. the cost of our developers, e.g.
    1. salaries of project team members
    2. costs of H/W and S/W
    3. overhead costs
2. **external**, i.e. the price to the client, e.g.
    1. usually internal costs plus some mark-up

Estimating Duration

The client will need to know when to expect the S/W product to be delivered.

Obstacles to Estimating Accurately

1. **human**
   1. variations in quality
   2. turnover
   3. varying levels of experience

Function Points

1. **Function Points** provide a consistent basis for comparing the sizes of different S/W products.
2. Some larger projects were counted in terms of **function points (FP)** during my time at SLF.

Function Points

Example like on pp273–275 in the text:

- Compute the **unadjusted function points (UFP)** for a software product having the following function point counts in conjunction with Figure 9.3 in the text (reproduced here).

Function Points

Figure 9.3 - Table of Function Point Values

	Level of Complexity		
Component	Simple	Average	Complex
Input item	3	4	6
Output item	4	5	7
Inquiry	3	4	6
Master file	7	10	15
Interface	5	7	10

Function Points

Function Point Counts to Use

	Level of Complexity		
Component	Simple	Average	Complex
Input item	12	8	0
Output item	10	7	1
Inquiry	8	4	1
Master file	1	1	1
Interface	6	2	0

Function Points

Solution:

$$
\begin{aligned}
\text{UFP(Input item)} &= (12)(3) + (8)(4) + (0)(6) \\
&= 68 \\
\text{UFP(Output item)} &= (10)(4) + (7)(5) + (1)(7) \\
&= 82 \\
\text{UFP(Inquiry)} &= (8)(3) + (4)(4) + (1)(6) \\
&= 46 \\
\text{UFP(Master file)} &= (1)(7) + (1)(10) + (1)(15) \\
&= 32 \\
\text{UFP(Interface)} &= (6)(5) + (2)(7) + (0)(10) \\
&= 44, \text{ so that the total UFP is} \\
\mathit{UFP} &= 68 + 82 + 46 + 32 + 44 \\
&= 272.
\end{aligned}
$$

Function Points

- Compute the **technical complexity factor (TCF)** using the given counts for each factor in Figure 9.4 from the text (reproduced here).

Function Points

Figure 9.4 (augmented) - Technical factors for function point computation

Factor	Name	Count to Use
1	Data communication	2
2	Distributed data processing	0
3	Performance criteria	3
4	Heavily utilized hardware	1
5	High transaction rates	3
6	Online data entry	5
7	End-user efficiency	5
8	Online updating	1
9	Complex computations	3
10	Reusability	3
11	Ease of installation	0
12	Ease of operation	5
13	Portability	3
14	Maintainability	5

Function Points

Solution: Summing the counts in the above table gives us the **total degree of influence**:

$$\begin{aligned} DI &= 2+0+3+1+3+5+5+1+3+3+0+5+3+5 \\ &= 39, \end{aligned}$$

so that the corresponding **technical complexity factor (TCF)** is

$$\begin{aligned} TCF &= 0.65 + (0.01)DI \\ &= 0.65 + (0.01)(39) \\ &= 1.04. \end{aligned}$$

- Use the results of the previous two parts to compute the **function points (FP)** for the given software product.

Solution:

$$\begin{aligned} FP &= (UFP)(TCF) \\ &= (272)(1.04) \\ &= 282.88, \end{aligned}$$

so that we measure this software product at $\boxed{283FP}$. (Only whole numbers make sense here; we **always round up** to be conservative.)

Remarks About Function Points

1. Observe that nowhere in the computation of UFP or FP did we ask
   1. in what language is this software product written? or
   2. how many lines of code does this software product have?

   FP are designed to be independent of these factors. FP compare sizes of different software products, regardless of their implementations.