CSSE 372 Software Project Management:
Software Estimation With COCOMO-II

Shawn Bohner
Office: Moench Room F212
Phone: (812) 877-8685
Email: bohner@rose-hulman.edu

ROSE-HULMAN
INSTITUTE OF TECHNOLOGY

Estimation Experience and Beware of the Sales Guy…

Cost Estimation

- Project scope must be explicitly defined
- Task, functional, or component decomposition is necessary
- Historical measures (metrics) are very helpful
- To assure fit, use two or more estimation techniques

- Uncertainty is inherent in most estimation endeavors – plan on it!

Learning Outcomes: Estimation

Estimate software project effort, cost, and schedule for an intermediate size project.

- Outline the COCOMO-II parametric model as an example
- Examine the key factors and adjustments
- Walk through getting started…

Constructive Cost Model (COCOMO)

- Empirical model based on project experience
- Well-documented, independent model, Independent of a specific software vendor
- Long history – initially published in 1981 (COCOMO-81) and last in 1999 (COCOMO-II)
- COCOMO-II takes into account different approaches to software development, reuse, etc.

Accommodating Progression

Three-level model that allows increasingly detailed estimates to be prepared as development progresses

- **Early prototyping level**
  - Based on "object points"
  - Simple formula
- **Early design level**
  - Based on "function points" that are then translated to lines of source code (LOC)
- **Post-architecture level**
  - Estimates based on LOC

Early Prototyping Level

- Supports prototyping projects and projects where there is extensive reuse
- Estimates effort in object points/staff month
- PM = (NOP × (1 - %reuse/100)) / PROD, where:
  - PM is the effort in person-months
  - NOP is the number of object points
  - PROD is the productivity

Early Design Level

- Estimates made after requirements confirmed

$$PM = A \times Size^{B} \times M + \prod_{I=1}^{n} EM_i$$

where:

- A = 2.5 in initial calibration
- Size in KLOC
- B varies from 1.1 to 1.24 depending on novelty of project, development flexibility, risk management approaches, and process maturity
- EM = (ASLOC × (AT / 100)) / ATPROD
- M = PERS × RCPX × RUSE × PDIF × PREX × FCIL × SCED

Multipliers

Multipliers reflect capability of developers, non-functional requirements, familiarity with development platform, etc.

RCPX - product reliability and complexity

RUSE - the reuse required

PDIF - platform difficulty

PREX - personnel experience

PERS - personnel capability

SCED - required schedule

FCIL - the team support facilities

Post-Architecture Level

- Uses same formula as early design estimates

- Estimate of size adjusted to account for:
  - Requirements volatility
  - Rework required to support change
  - Extent of possible reuse

Post-Architecture Level (continued)

$$ESLOC = ASLOC \times (AA + SU + 0.4DM + 0.3CM + 0.3IM) / 100$$

- ESLOC is equivalent number of lines of new code
- ASLOC is the adjusted number of lines of reusable code which must be modified
- DM is the % of design modified
- CM is the % of the code that is modified
- IM is the % of the original integration effort required for integrating the reused software
- SU is a factor based on the cost of software understanding
- AA is a factor which reflects the initial assessment costs of deciding if software may be reused

Exponent Scale Factors

Scale factor	Explanation
Precedentedness	Reflects the previous experience of the organization with this type of project. Very low means no previous experience, Extra high means that the organization is completely familiar with this application domain.
Development flexibility	Reflects the degree of flexibility in the development process. Very low means a prescribed process is used; Extra high means that the client only sets general goals.
Architecture/risk resolution	Reflects the extent of risk analysis carried out. Very low means little analysis, Extra high means a complete a thorough risk analysis.
Team cohesion	Reflects how well the development team know each other and work together. Very low means very difficult interactions, Extra high means an integrated and effective team with no communication problems.
Process maturity	Reflects the process maturity of the organization. The computation of this value depends on the CMM Maturity Questionnaire but an estimate can be achieved by subtracting the CMM process maturity level from 5.

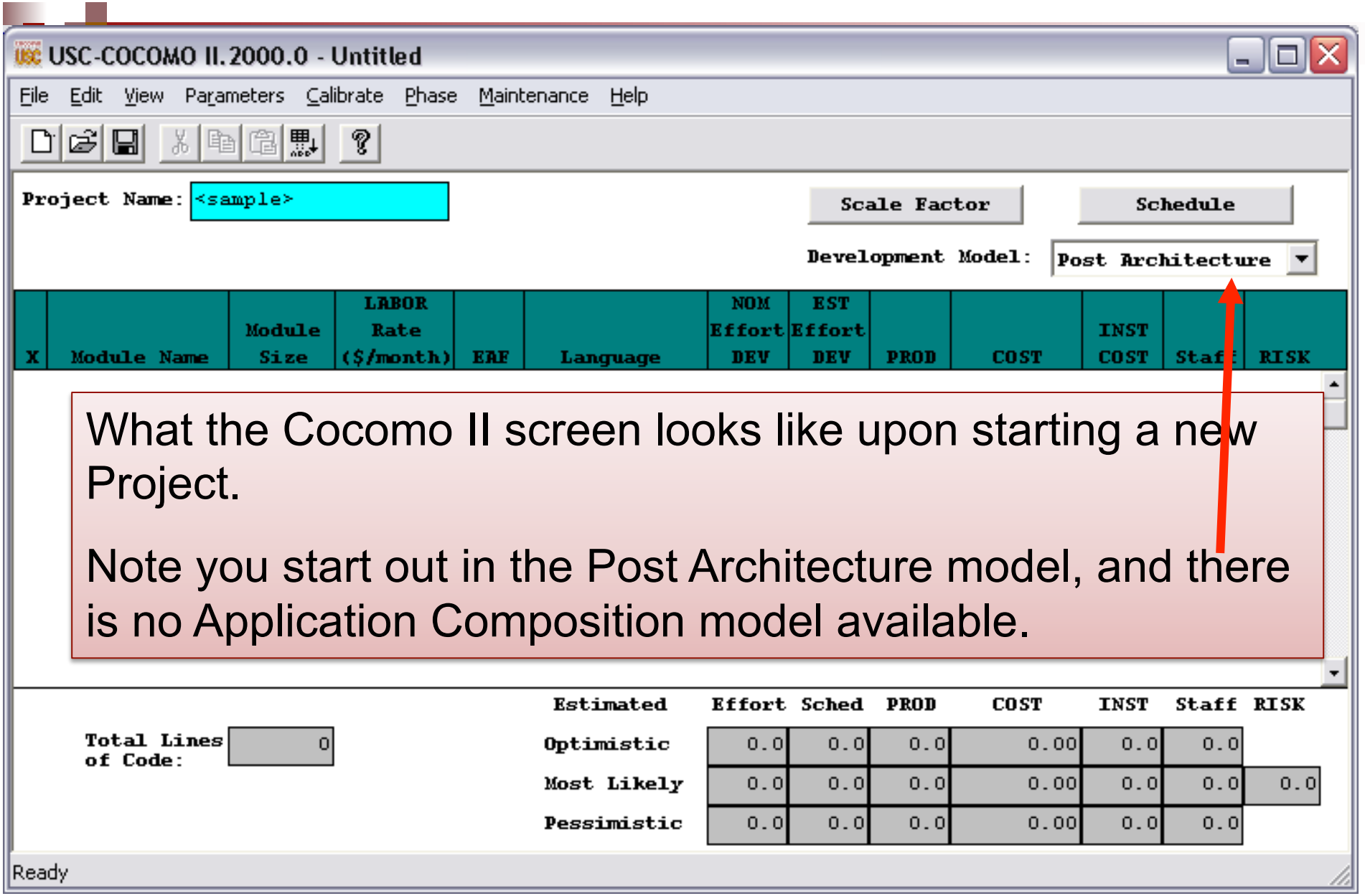
Project Cost Drivers

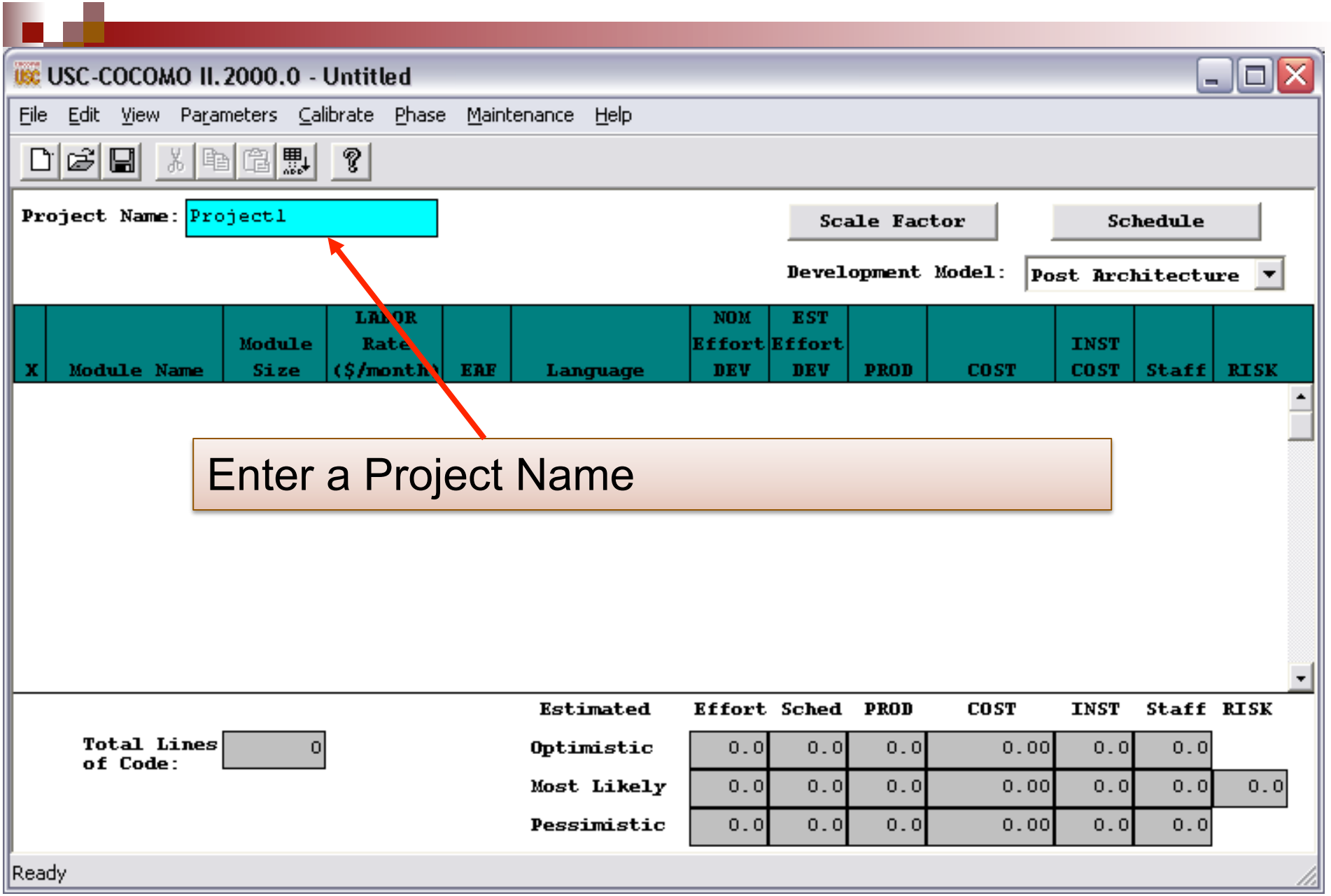
Product attributes			
RELY	Required system reliability	DATA	Size of database used
CPLX	Complexity of system modules	RUSE	Required percentage of reusable components
DOCU	Extent of documentation required		
Computer attributes			
TIME	Execution time constraints	STOR	Memory constraints
PVOL	Volatility of development platform		
Personnel attributes			
ACAP	Capability of project analysts	PCAP	Programmer capability
PCON	Personnel continuity	AEXP	Analyst experience in project domain
PEXP	Programmer experience in project domain	LTEX	Language and tool experience
Project attributes			
TOOL	Use of software tools	SITE	Extent of multi-site working and quality of site communications
SCED	Development schedule compression		

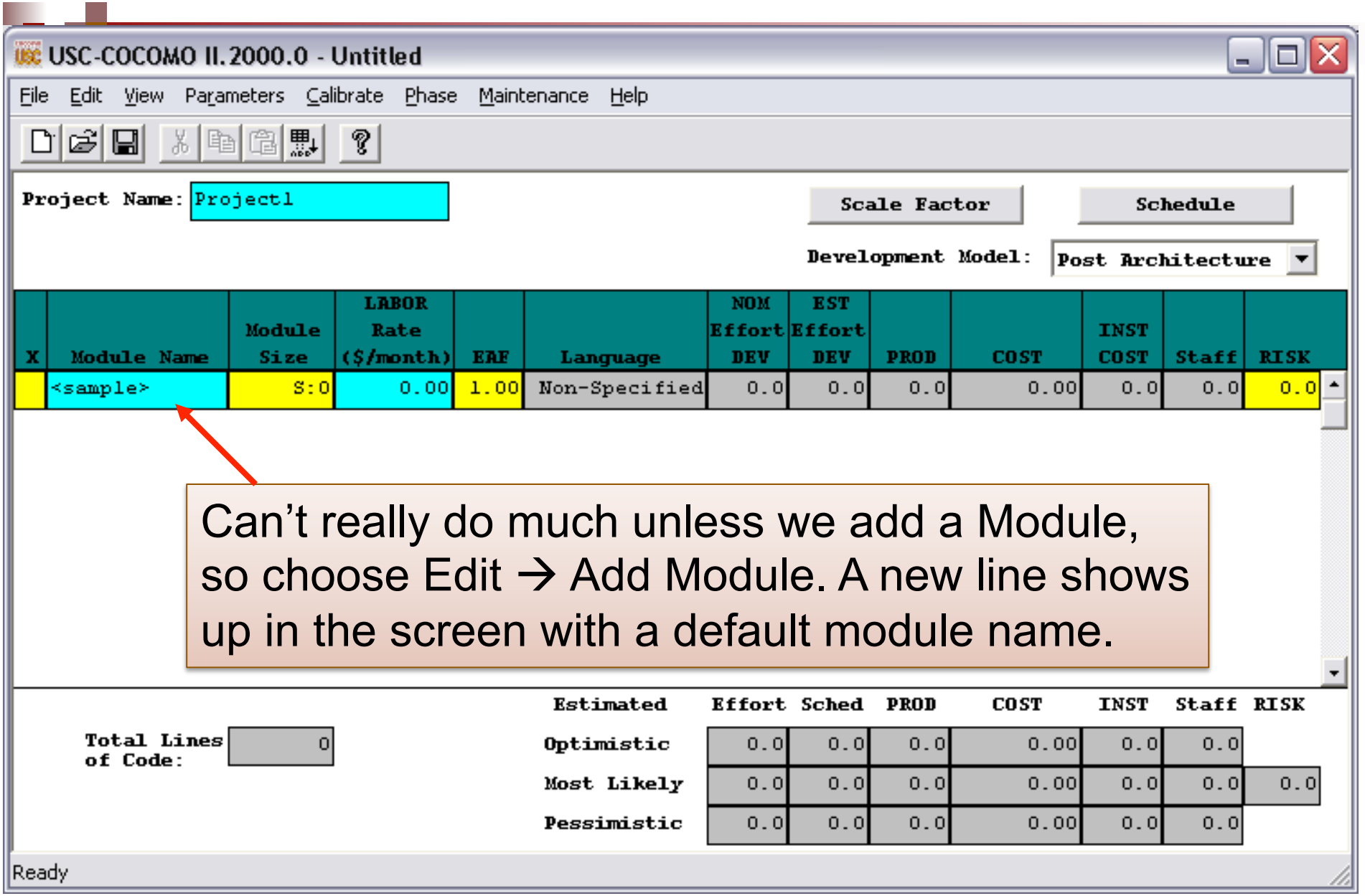
ESTIMATION

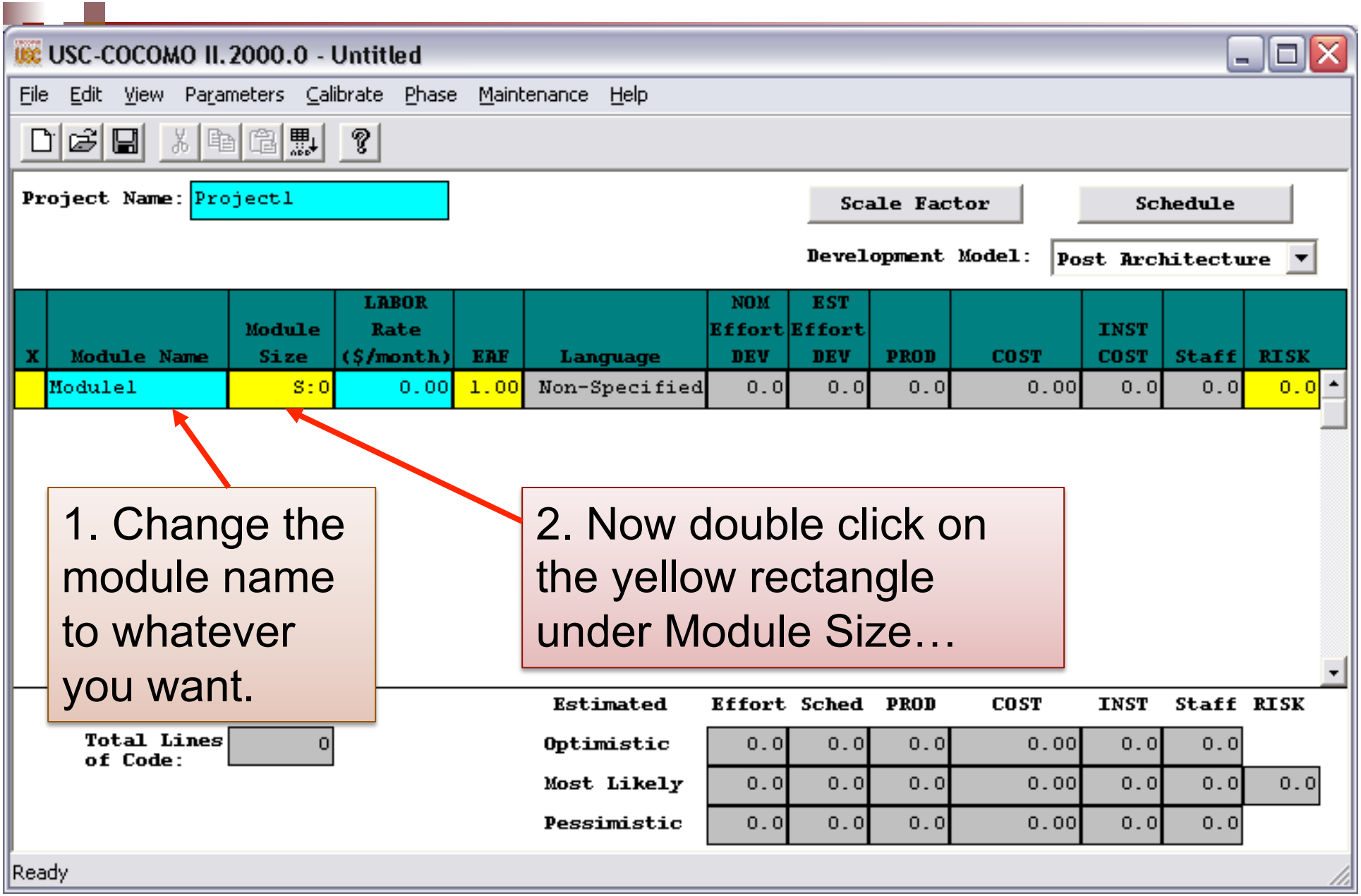
GOOD ESTIMATORS DO NOT EXAGGERATE.
IF IT'S HUGE, THEY SAY SO. YOU SHOULD BELIEVE THEM.

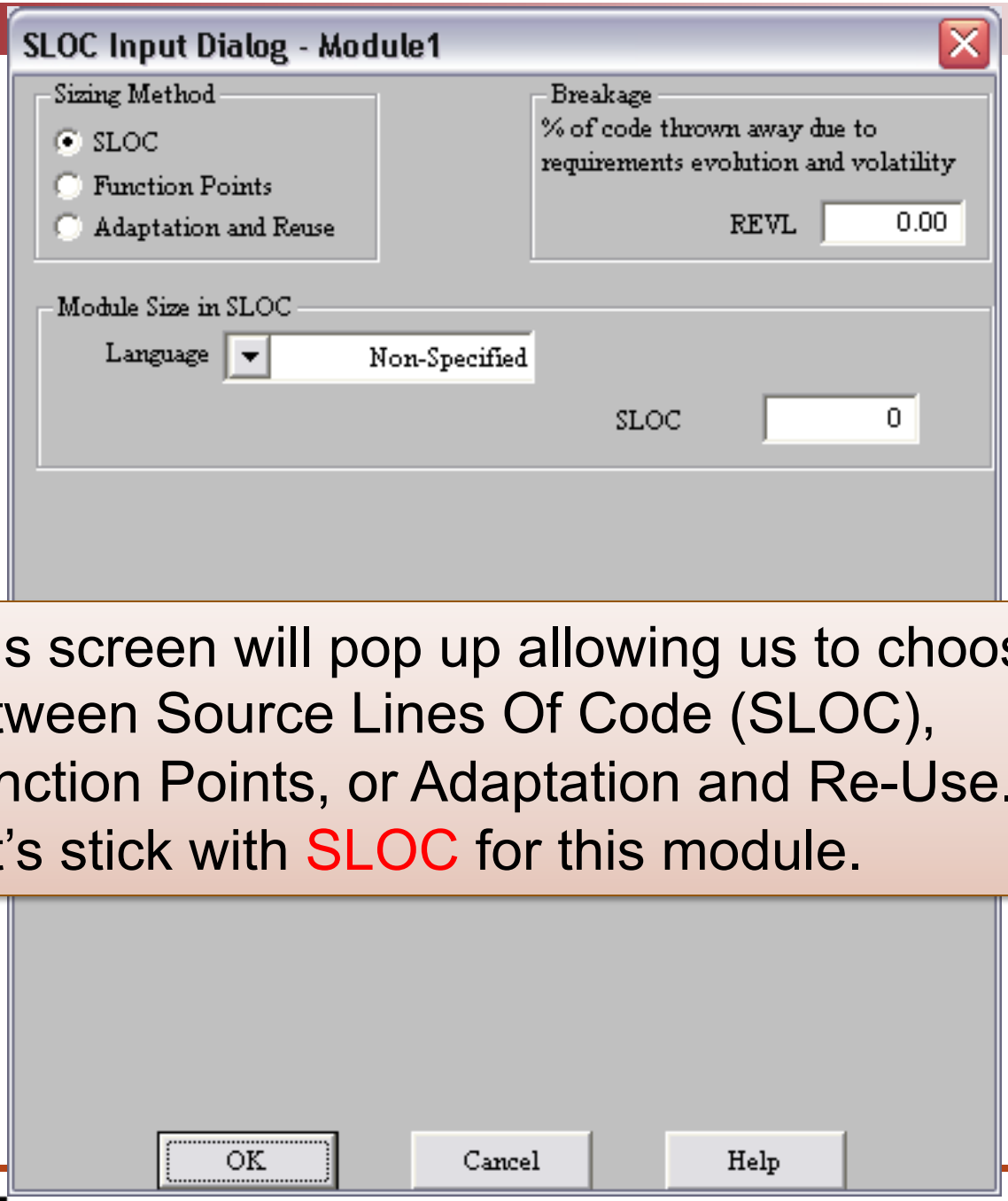
motifake.com


USC-COCOMO II.2000.0 - Untitled
File Edit View Parameters Calibrate Phase Maintenance Help
Project Name: <sample>
Scale Factor
Schedule
Development Model: Post Architecture
X Module Name Module Size LABOR Rate ($/month) EAF Language NOM Effort DEV EST Effort DEV PROD COST INST COST Staff RISK
What the Cocomo II screen looks like upon starting a new Project.
Note you start out in the Post Architecture model, and there is no Application Composition model available.
Total Lines of Code: 0
Estimated Effort Sched PROD COST INST Staff RISK
Optimistic 0.0 0.0 0.0 0.00 0.0 0.0
Most Likely 0.0 0.0 0.0 0.00 0.0 0.0 0.0
Pessimistic 0.0 0.0 0.0 0.00 0.0 0.0
Ready

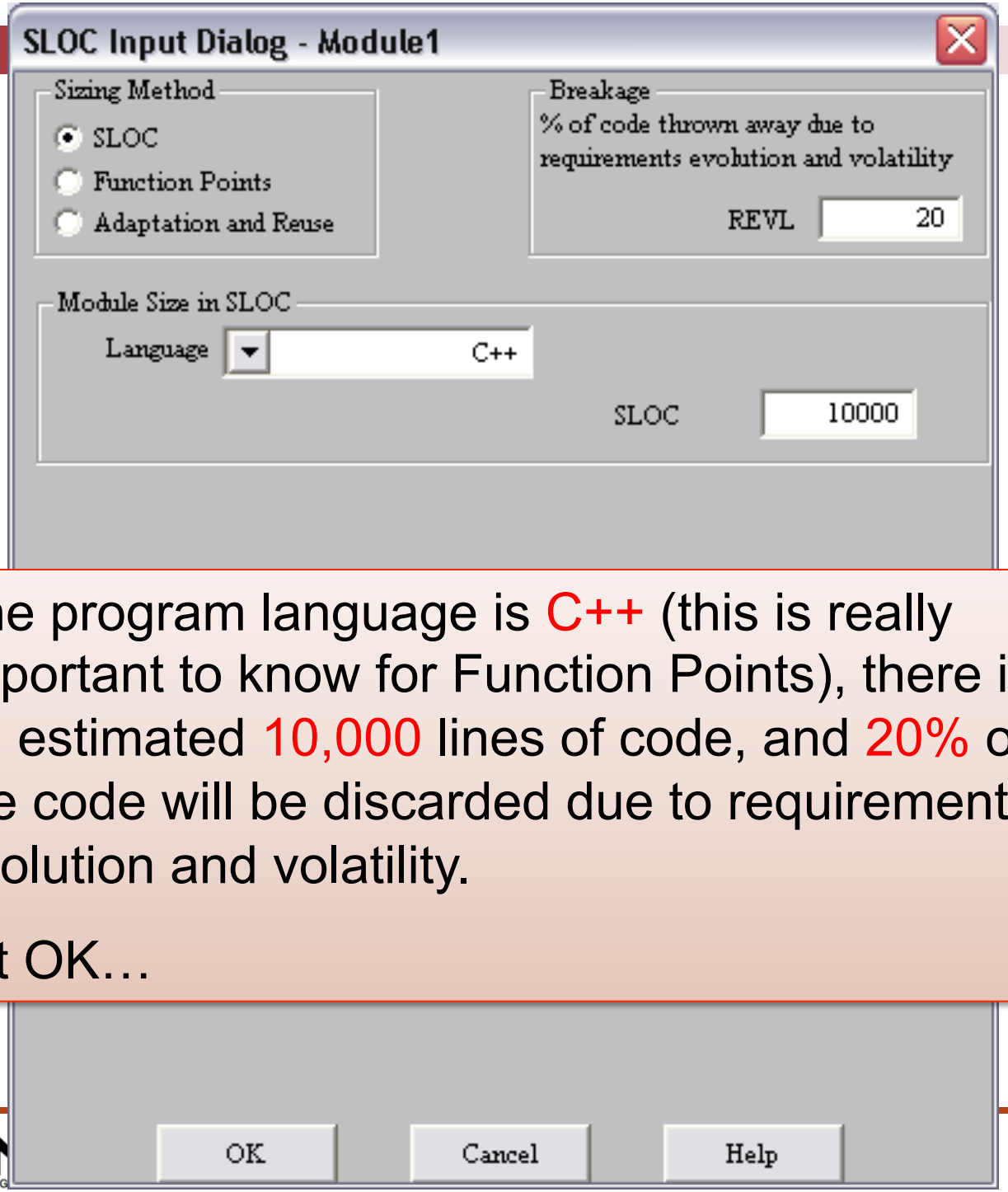

USC-COCOMO II.2000.0 - Untitled
File Edit View Parameters Calibrate Phase Maintenance Help
Project Name: Project1
Scale Factor
Schedule
Development Model: Post Architecture
X Module Name Module Size LABOR Rate ($/month) EAF Language NOM Effort DEV EST Effort DEV PROD COST INST COST Staff RISK
Enter a Project Name
Total Lines of Code: 0
Estimated Effort Sched PROD COST INST Staff RISK
Optimistic 0.0 0.0 0.0 0.00 0.0 0.0
Most Likely 0.0 0.0 0.0 0.00 0.0 0.0 0.0
Pessimistic 0.0 0.0 0.0 0.00 0.0 0.0
Ready


USC-COCOMO II.2000.0 - Untitled
File Edit View Parameters Calibrate Phase Maintenance Help
Project Name: Project1
Scale Factor
Schedule
Development Model: Post Architecture
X | Module Name | Module Size | LABOR Rate ($/month) | EAF | Language | NOM Effort DEV | EST Effort DEV | PROD | COST | INST COST | Staff | RISK
<sample> S:0 0.00 1.00 Non-Specified 0.0 0.0 0.0 0.00 0.0 0.0 0.0
Can't really do much unless we add a Module, so choose Edit → Add Module. A new line shows up in the screen with a default module name.
Total Lines of Code: 0
Estimated Effort Sched PROD COST INST Staff RISK
Optimistic 0.0 0.0 0.0 0.00 0.0 0.0
Most Likely 0.0 0.0 0.0 0.00 0.0 0.0 0.0
Pessimistic 0.0 0.0 0.0 0.00 0.0 0.0
Ready


USC-COCOMO II.2000.0 - Untitled
File Edit View Parameters Calibrate Phase Maintenance Help
Project Name: Project1
Scale Factor
Schedule
Development Model: Post Architecture
X | Module Name | Module Size | LABOR Rate ($/month) | EAF | Language | NOM Effort DEV | EST Effort DEV | PROD | COST | INST COST | Staff | RISK
Module1 | S:0 | 0.00 | 1.00 | Non-Specified | 0.0 | 0.0 | 0.0 | 0.00 | 0.0 | 0.0 | 0.0
1. Change the module name to whatever you want.
2. Now double click on the yellow rectangle under Module Size…
Total Lines of Code: 0
Estimated | Effort | Sched | PROD | COST | INST | Staff | RISK
Optimistic | 0.0 | 0.0 | 0.0 | 0.00 | 0.0 | 0.0
Most Likely | 0.0 | 0.0 | 0.0 | 0.00 | 0.0 | 0.0 | 0.0
Pessimistic | 0.0 | 0.0 | 0.0 | 0.00 | 0.0 | 0.0
Ready

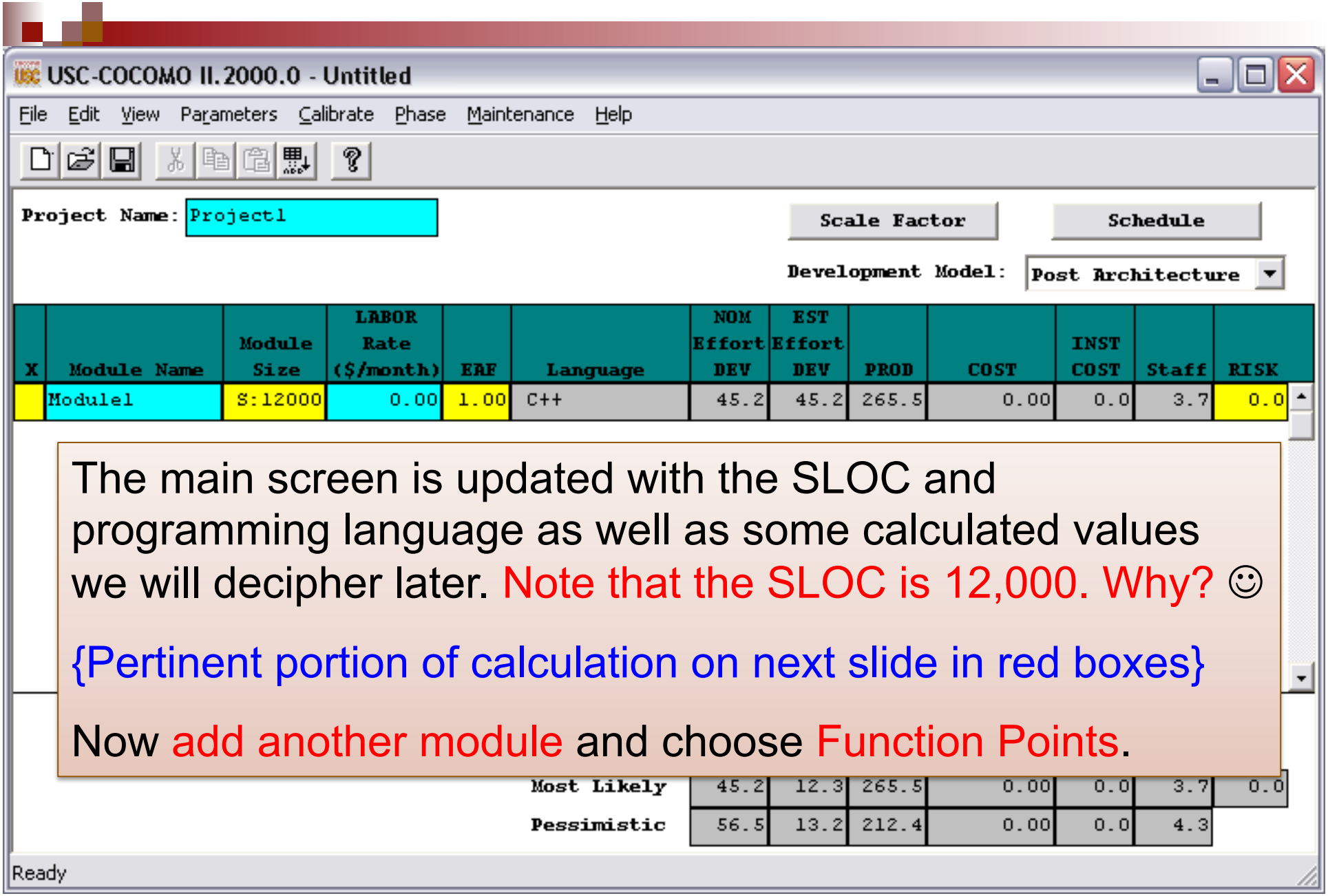


This screen will pop up allowing us to choose between Source Lines Of Code (SLOC), Function Points, or Adaptation and Re-Use. Let's stick with SLOC for this module.



The program language is C++ (this is really important to know for Function Points), there is an estimated 10,000 lines of code, and 20% of the code will be discarded due to requirements evolution and volatility.

Hit OK…



The main screen is updated with the SLOC and programming language as well as some calculated values we will decipher later. Note that the SLOC is 12,000. Why? ☺

{Pertinent portion of calculation on next slide in red boxes}

Now add another module and choose Function Points.

COCOMO-II Calculations: Effort Equation for Post Architecture Model

$$PM = \prod_{i=1}^{17} (EM_i) \cdot A \cdot \left[\left(1 + \frac{\text{REVL}}{100}\right) \cdot \text{Size}\right]^{\left(0.91 + 0.01 \sum_{j=1}^{5} SF_j\right)} + \left(\frac{\text{ASLOC} \cdot \left(\frac{AT}{100}\right)}{ATPROD}\right)$$

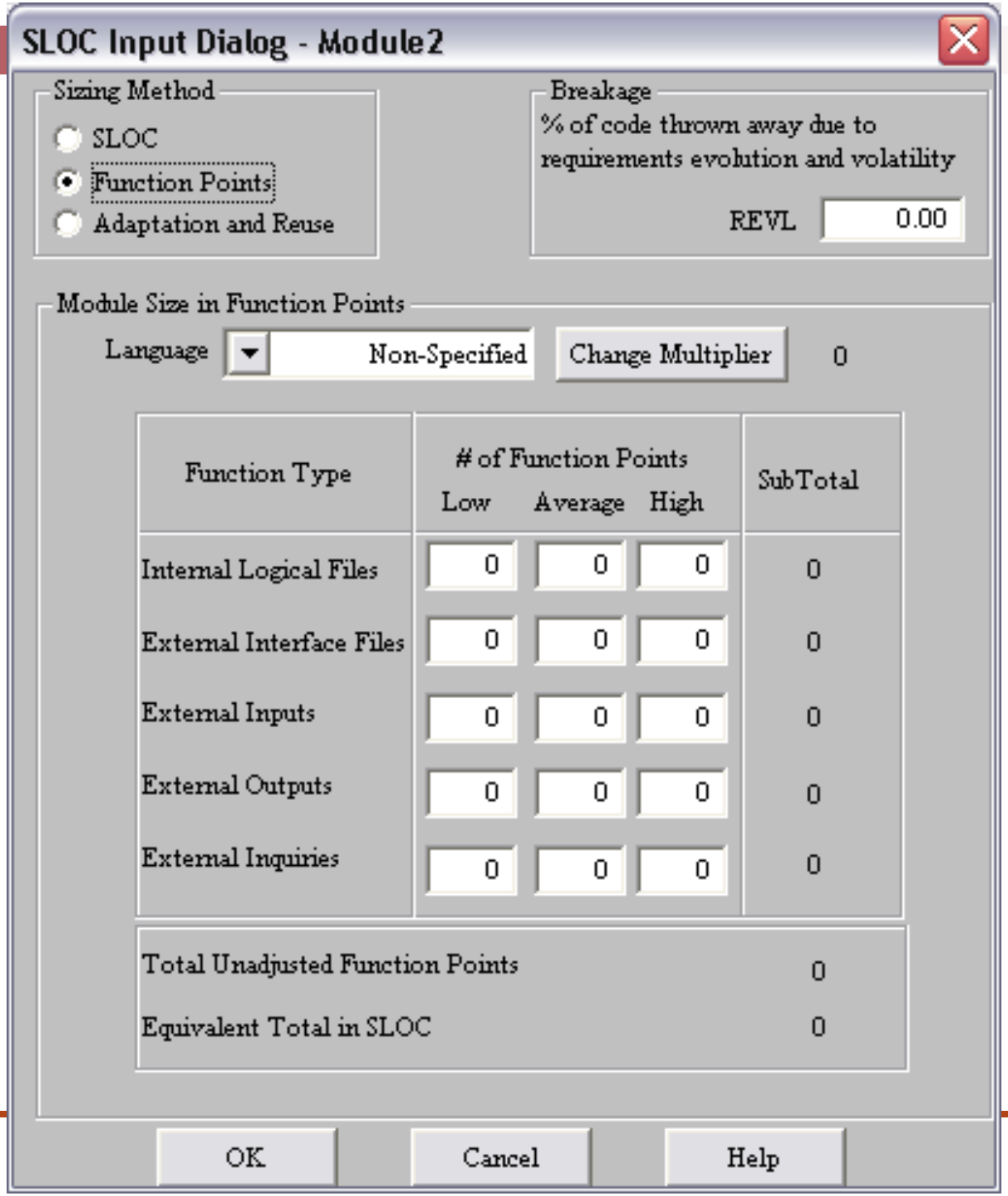
where

$$\text{Size} = \text{KNSLOC} + \left[\text{KASLOC} \cdot \left(\frac{100 - AT}{100}\right) \cdot \frac{(AA + SU + 0.4 \cdot DM + 0.3 \cdot CM + 0.3 \cdot IM)}{100}\right]$$

$$B = 0.91 + 0.01 \sum_{j=1}^{5} SF$$

This is the default screen for Function Points.

Let's look deeper at the Function Type descriptions…



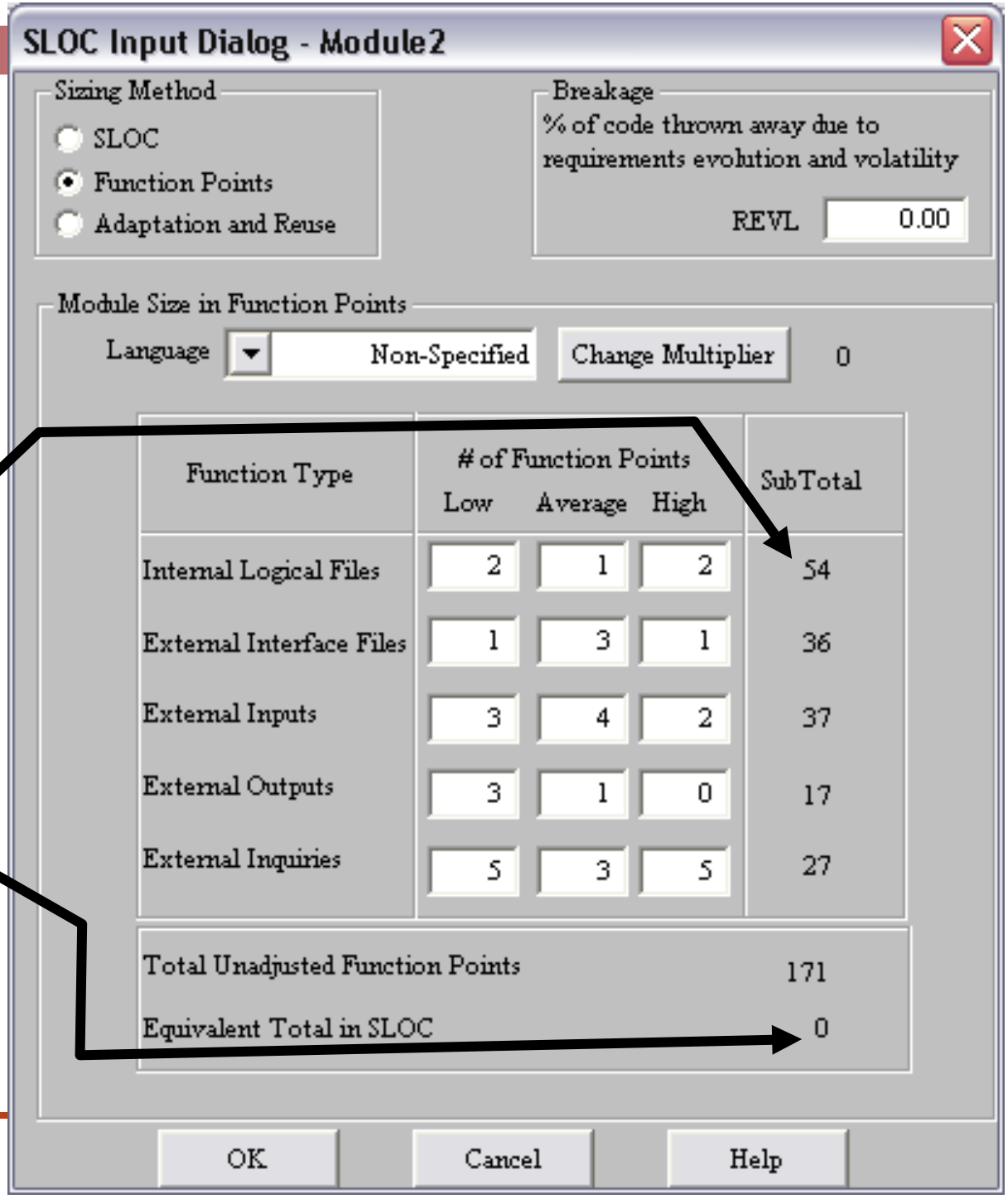
External Input (Inputs)	Count each unique user data or user control input type that (i) enters the external boundary of the software system being measured and (ii) adds or changes data in a logical internal file.
External Output (Outputs)	Count each unique user data or control output type that leaves the external boundary of the software system being measured.
Internal Logical File (Files)	Count each major logical group of user data or control information in the software system as a logical internal file type. Include each logical file (e.g., each logical group of data) that is generated, used, or maintained by the software system.
External Interface Files (Interfaces)	Files passed or shared between software systems should be counted as external interface file types within each system.
External Inquiry (Queries)	Count each unique input-output combination, where an input causes and generates an immediate output, as an external inquiry type.

From Cocomo II User Manual via software

So let's go back into this screen and add some entries in the grid.

Notice, there are some kind of subtotals per line, but the Equivalent SLOC = 0.

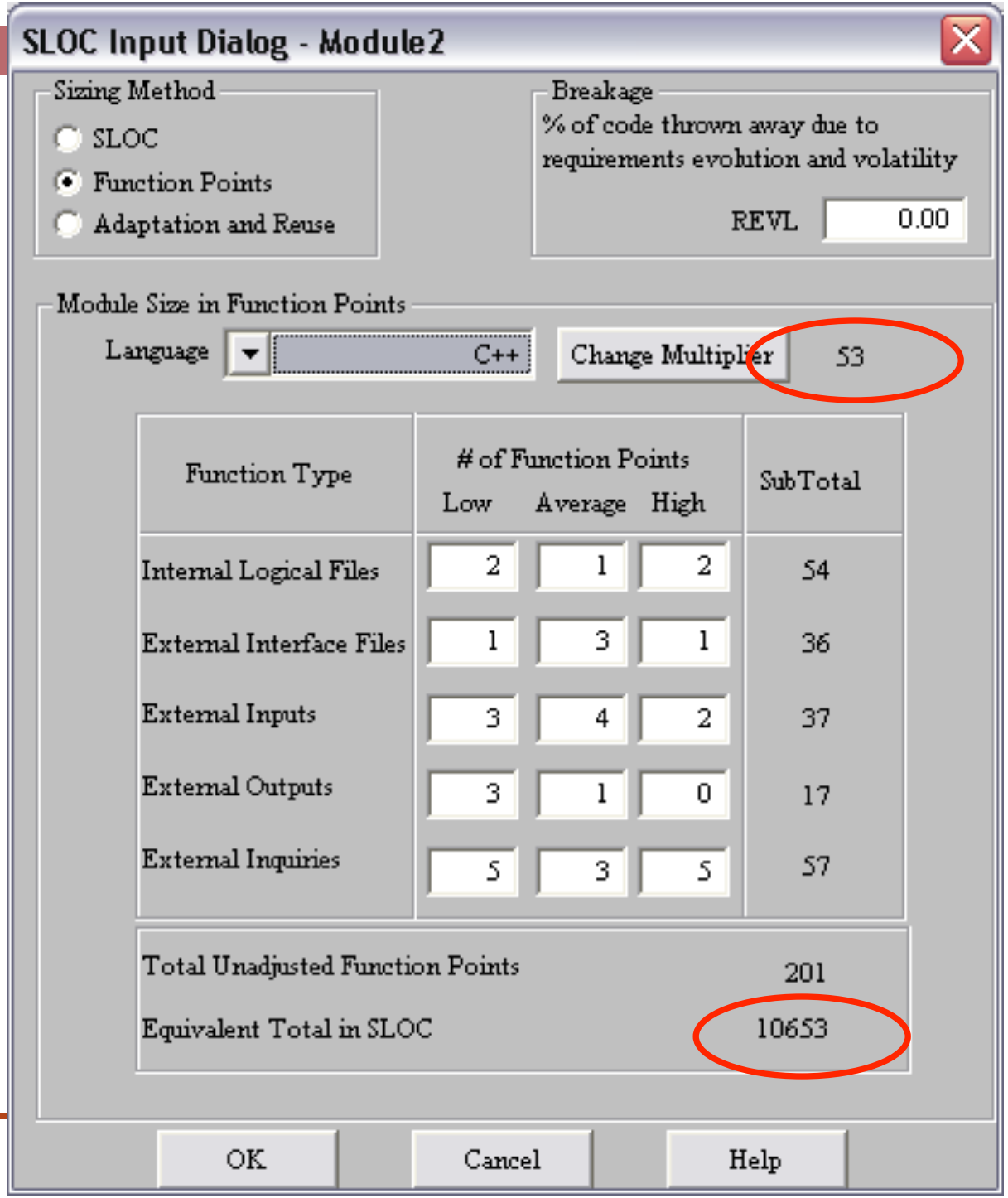
Let's change the Language and see what happens.



By changing the language to C++, we now have an Equivalent Total in SLOC.

Also, we can see a value next to the Change Multiplier button.

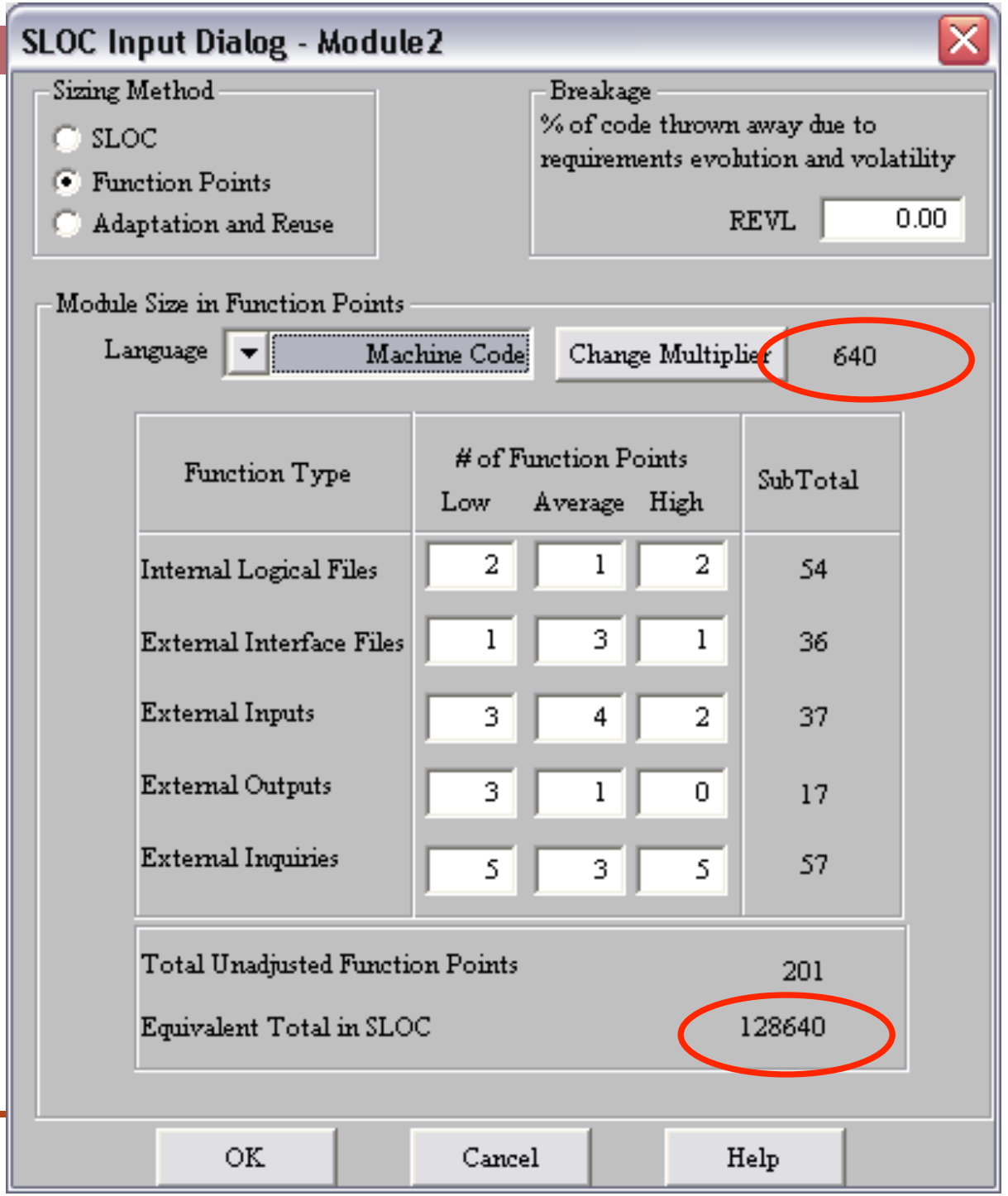
Let's change the language to Machine Code! ☺



Quite a difference jumping from 10,653 SLOC to 128,640 SLOC.

Note the multiplier changed from 53 to 640.

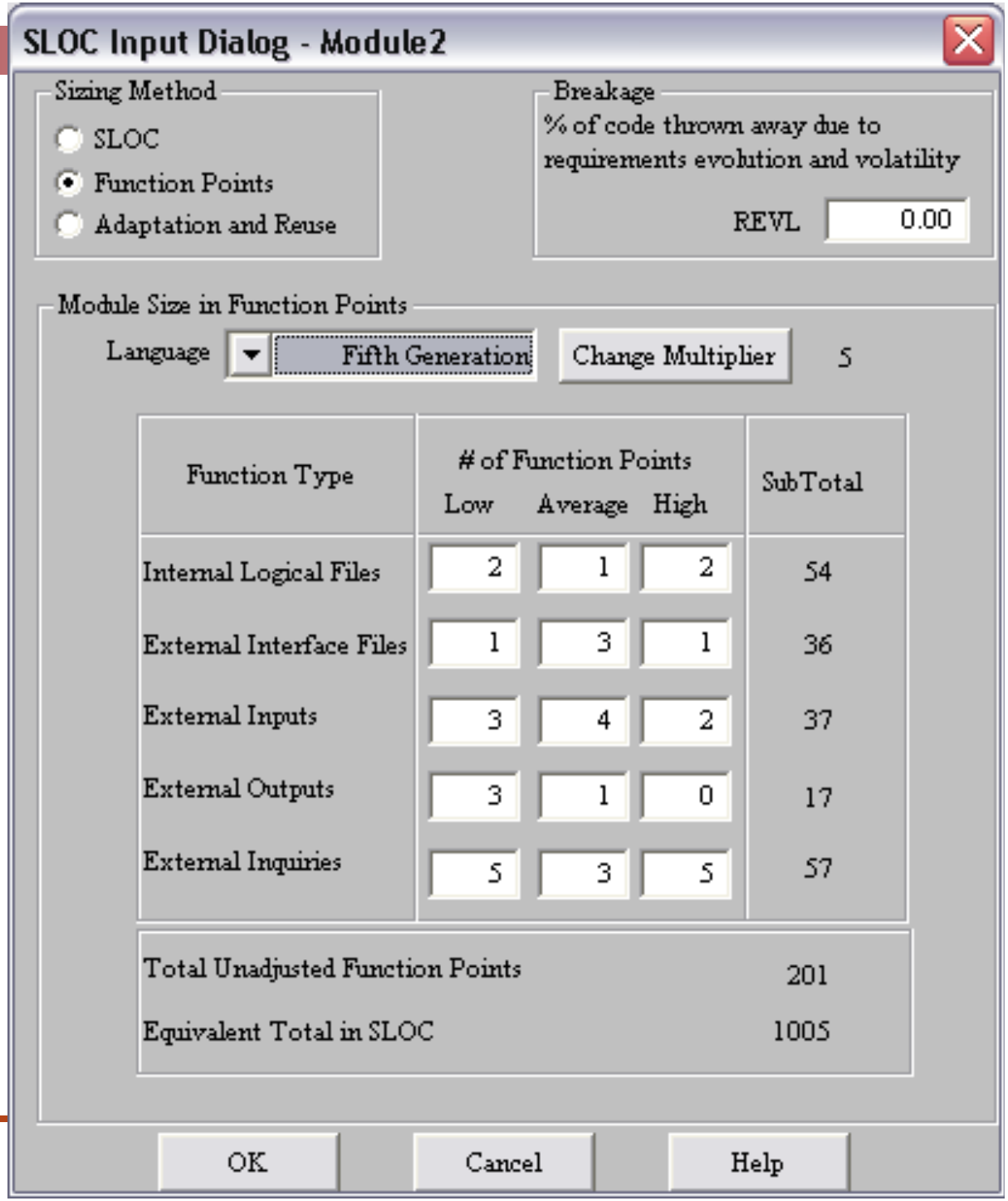
Change the language once more to 5th Generation.



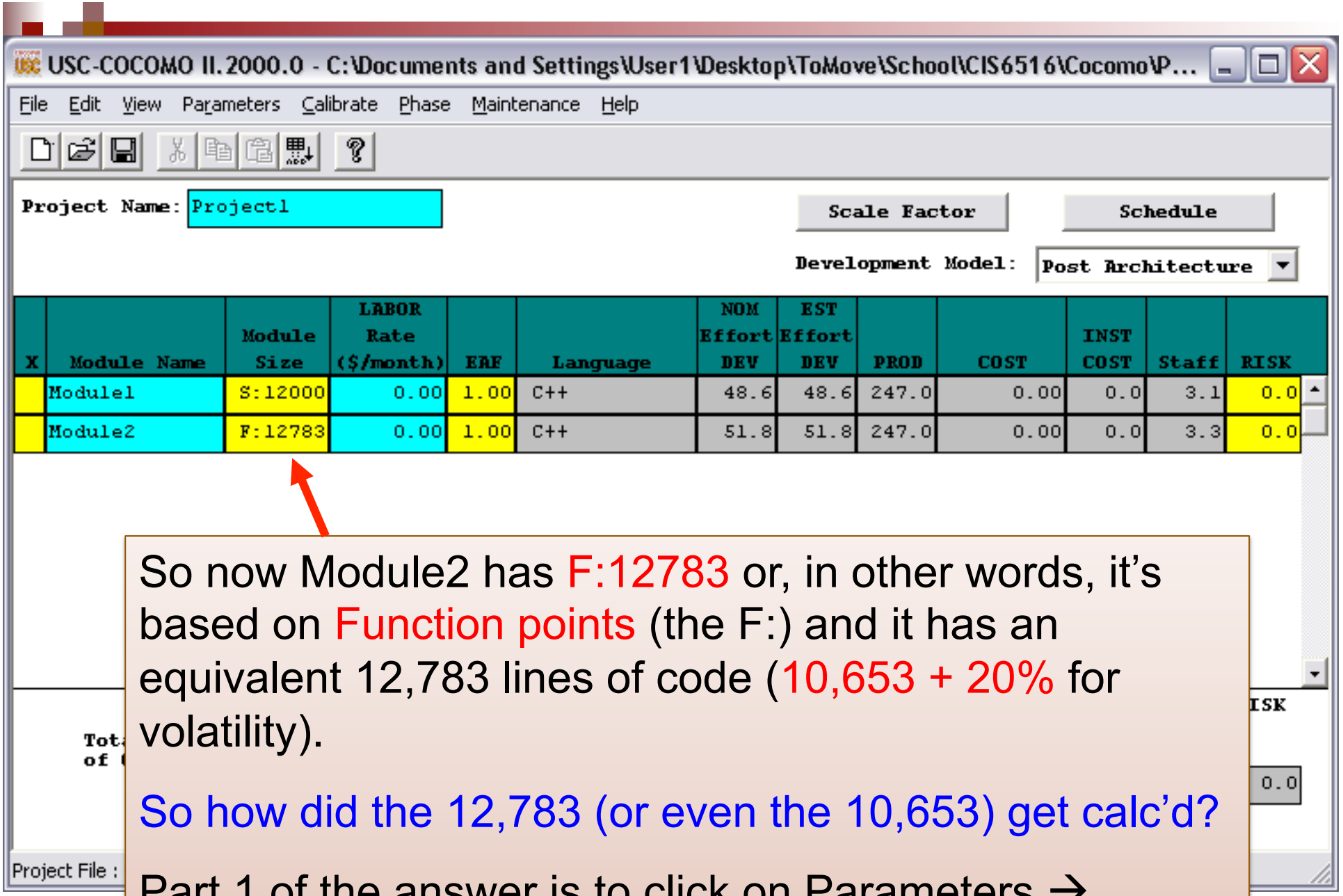
ROSE-HULMAN INSTITUTE OF TECHNOLOGY

So using a 5th generation level language would cut our code base by a factor of 285 times according to COCOMO II's default estimation (not calibrated for your environment, not taking into account other factors).

Change the language to C++ and change REVL to 20%…



ROSE-HULMAN INSTITUTE OF TECHNOLOGY



So now Module2 has F:12783 or, in other words, it's based on Function points (the F:) and it has an equivalent 12,783 lines of code (10,653 + 20% for volatility).

So how did the 12,783 (or even the 10,653) get calc'd?

Part 1 of the answer is to click on Parameters → Function Points. You will see the following screen…

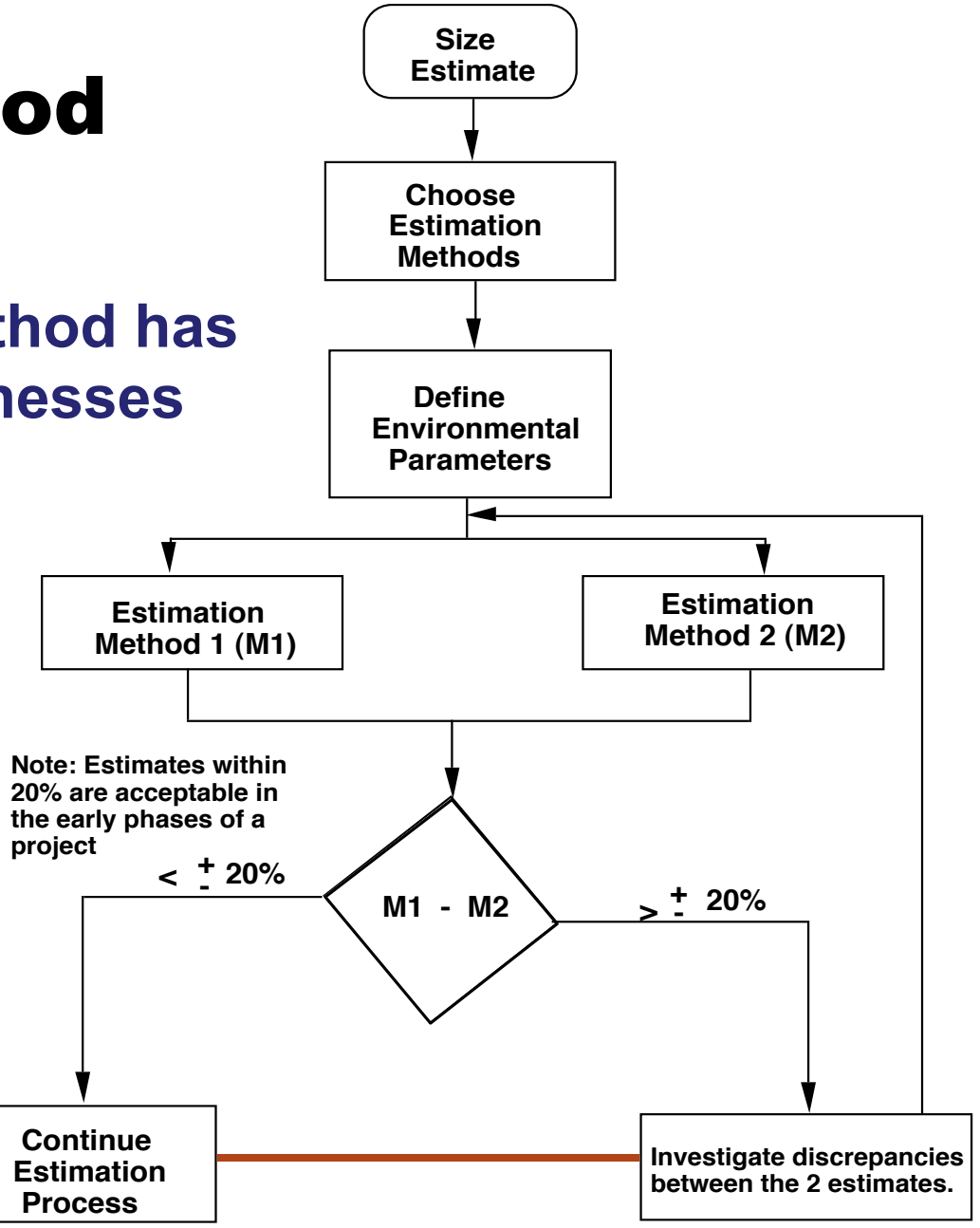
Function Point - Default model values used

Function Type	Low	Average	High
Internal Logical Files	7	10	15
External Interface Files	5	7	10
External Inputs	3	4	6
External Outputs	4	5	7
External Inquiries	3	4	6

These are the default values used as weighting factors against the entries you put in. So if you entered 2,3,4 when enter in Function Point information for the first row, the end result would be 2*7 + 3*10 + 4*15. This is then multiplied by The Change Multiplier…

Triangulation on Estimation Method Results

Each Estimation Method has strengths and weaknesses need to offset.



Homework and Reading Reminders

- Read Paper for Thursday's Case Study
- Complete Homework 3 – Software Estimate Using COCOMO-II or Costar
  - Due by 5pm, Tuesday, September 25th, 2012