COCOMO

Lecture in the
CS6022 – Software Project Management

Dr.R.BASKARAN
Professor
baaski@annauniv.edu

Department of Computer Science and Engineering,
Anna University, Chennai – 600 025.

COST OF A PROJECT

Department of Computer Science and Engineering,
Anna University, Chennai – 600 025.

- The cost in a project is due to:
  - the requirements for software, hardware and human resources
  - the cost of software development is due to the human resources needed
  - most cost estimates are measured in *person-months (PM)*
  - the cost of the project depends on the nature and characteristics of the project,
  - at any point, the accuracy of the estimate will depend on the amount of reliable information we have about the final product.

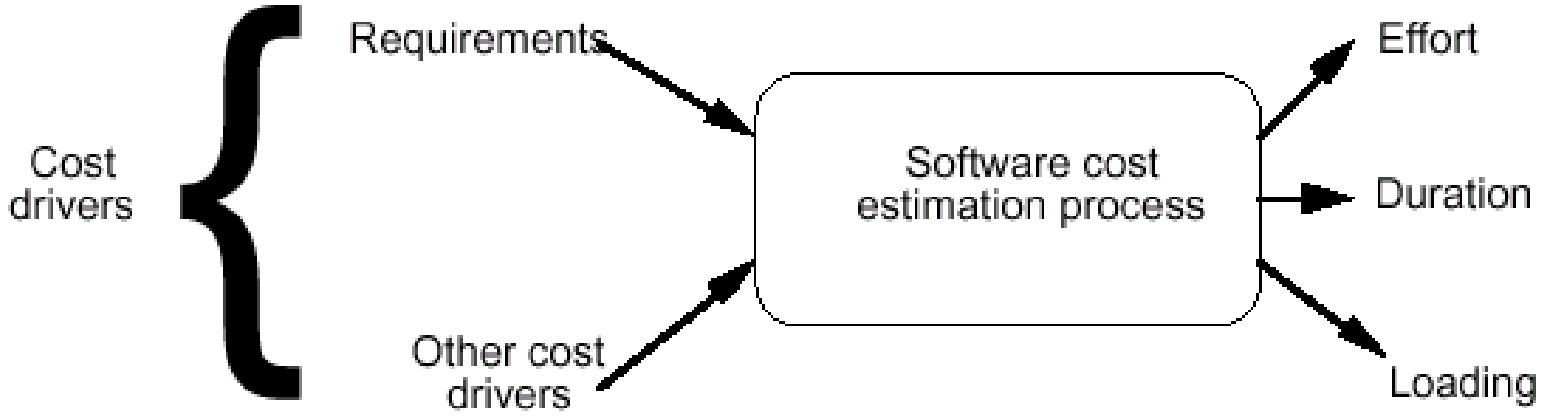


Figure 1. Classical view of software estimation process.

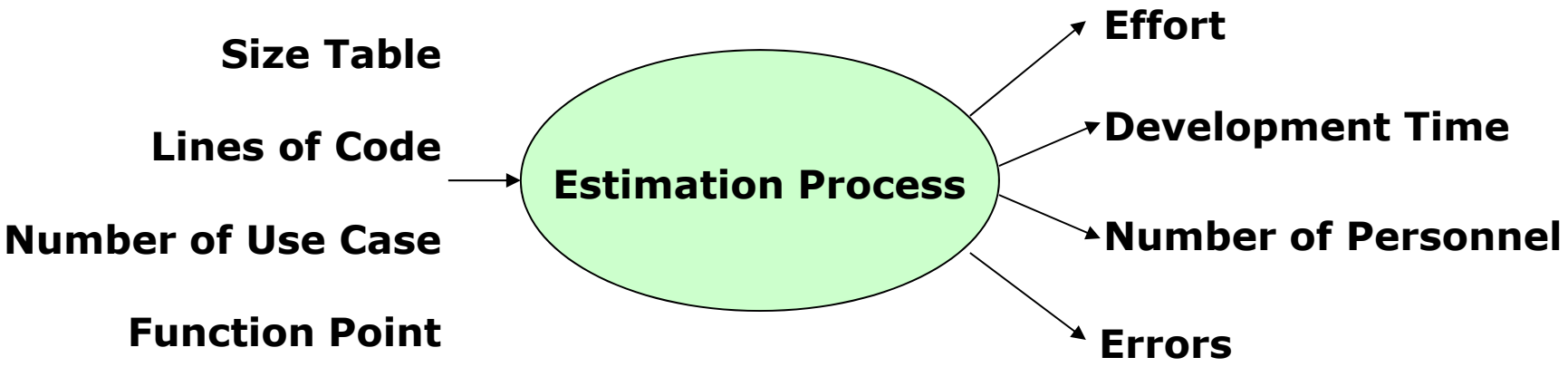
EFFORT

Department of Computer Science and Engineering,
Anna University, Chennai – 600 025.

- Effort Equation
  - **PM = C * (KDSI)^n (**person-months)
    - where **PM** = number of person-month (=152 working hours),
    - **C** = a constant,
    - **KDSI** = thousands of "delivered source instructions" (DSI) and
    - **n** = a constant.
- Productivity
  - **(DSI) / (PM)**
    - where **PM** = number of person-month (=152 working hours),
    - **DSI** = "delivered source instructions“
- Schedule equation
  - **TDEV = C * (PM)^n (**months)
    - where TDEV = number of months estimated for software development.
- Average Staffing Equation
  - **(PM) / (TDEV)** **(**FSP)
    - where FSP means Full-time-equivalent Software Personnel.

COST ESTIMATION PROCESS

Department of Computer Science and Engineering,
Anna University, Chennai – 600 025.



PROJECT SIZE - METRICS

Department of Computer Science and Engineering,
Anna University, Chennai – 600 025.

1. Number of functional requirements
2. Cumulative number of functional and non-functional requirements
3. Number of Customer Test Cases
4. Number of 'typical sized' use cases
5. Number of inquiries
6. Number of files accessed (external, internal, master)
7. Total number of components (subsystems, modules, procedures, routines, classes, methods)
8. Total number of interfaces
9. Number of System Integration Test Cases
10. Number of input and output parameters (summed over each interface)
11. Number of Designer Unit Test Cases
12. Number of decisions (if, case statements) summed over each routine or method
13. **Lines of Code, summed over each routine or method**

Availability of Size Estimation Metrics:

	Development Phase	Available Metrics
a	Requirements Gathering	1, 2, 3
b	Requirements Analysis	4, 5
d	High Level Design	6, 7, 8, 9
e	Detailed Design	10, 11, 12
f	Implementation	12, 13

FUNCTION POINTS

Department of Computer Science and Engineering,
Anna University, Chennai – 600 025.

STEP 1: measure size in terms of the amount of functionality in a system. Function points are computed by first calculating an ***unadjusted function point count*** **(UFC).** Counts are made for the following categories

- *External inputs* – those items provided by the user that describe distinct application-oriented data (such as file names and menu selections)
- *External outputs* – those items provided to the user that generate distinct application-oriented data (such as reports and messages, rather than the individual components of these)
- *External inquiries* – interactive inputs requiring a response
- *External files* – machine-readable interfaces to other systems
- *Internal files* – logical master files in the system

STEP 2: Multiply each number by a weight factor, according to complexity (**simple**, **average** or **complex**) of the parameter, associated with that number. The value is given by a table:

Parameter	simple	average	complex
users inputs	3	4	6
users outputs	4	5	7
users requests	3	4	6
files	7	10	15
external interfaces	5	7	10

STEP 3: Calculate the total **UFP** (Unadjusted Function Points)
STEP 4: Calculate the total **TCF** (Technical Complexity Factor) by giving a value between 0 and 5 according to the importance of the following points

STEP 5: Sum the resulting numbers too obtain **DI** (degree of influence)
STEP 6: TCF (Technical Complexity Factor) by given by the formula

TCF=0.65+0.01*DI

STEP 7: Function Points are by given by the formula

FP=UFP*TCF

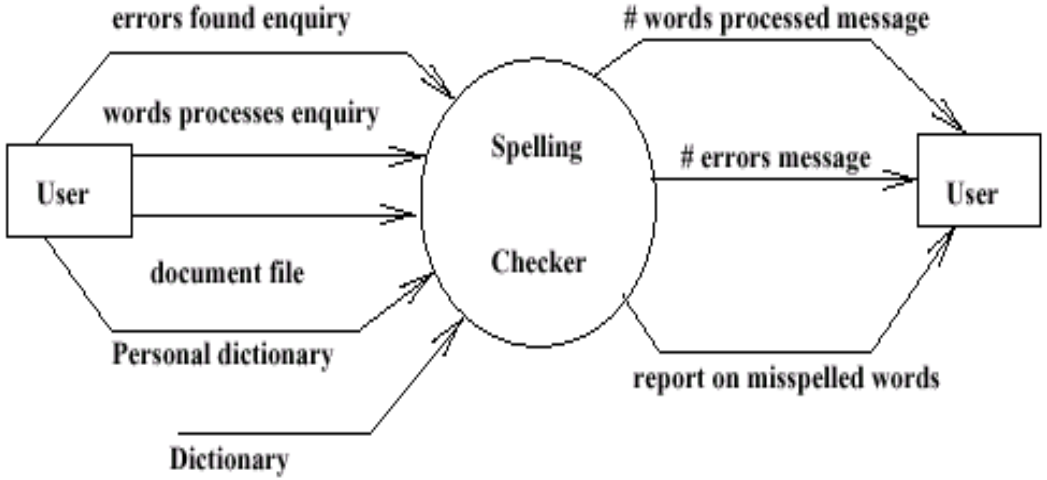
Technical Complexity Factors

1. Data Communication
2. Distributed Data Processing
3. Performance Criteria
4. Heavily Utilized Hardware
5. High Transaction Rates
6. Online Data Entry
7. Online Updating
8. End-user Efficiency
9. Complex Computations
10. Reusability
11. Ease of Installation
12. Ease of Operation
13. Portability
14. Maintainability

Department of Computer Science and Engineering,
Anna University, Chennai – 600 025.

Example

The Spell-Checker accepts as input a document file and an optional personal dictionary file. The checker lists all words not contained in either of these files. The user can query the number of words processed and the number of spelling errors found at any stage during processing.



- 2 users inputs: document file name, personal dictionary name (average)
- 3 users outputs: fault report, word count, misspelled error count (average)
- 2 users requests: #treated words?, #found errors? (average)
- 1 internal file: dictionary (average)
- 2 external files: document file, personal dictionary (av).

$$UFP = 4 \times 2 + 5 \times 3 + 4 \times 2 + 10 \times 1 + 7 \times 2 = 55$$

- **Technical Complexity Factors:**
  - 1. Data Communication 3
  - 2. Distributed Data Processing 0
  - 3. Performance Criteria 4
  - 4. Heavily Utilized Hardware 0
  - 5. High Transaction Rates 3
  - 6. Online Data Entry 3
  - 7. Online Updating 3
  - 8. End-user Efficiency 3
  - 9. Complex Computations 0
  - 10. Reusability 3
  - 11. Ease of Installation 3
  - 12. Ease of Operation 5
  - 13. Portability 3
  - 14. Maintainability 3
    - DI =30 (Degree of Influence)

- Function Points
  - FP=UFP*(0.65+0.01*DI)= 55*(0.65+0.01*30)=52.25
  - That means **FP=52.25**

RELATION BETWEEN LOC AND FP

Department of Computer Science and Engineering,
Anna University, Chennai – 600 025.

- Relationship:
  - ***LOC = Language Factor * FP***
  - where
    - **LOC** (Lines of Code)
    - **FP** (Function Points)

Assuming LOC's per FP for:

Java = 53,

C++ = 64

aKLOC = FP * LOC_per_FP / 1000

It means for the SpellChekcer Example: (Java)

LOC=52.25*53=2769.25 LOC or 2.76 KLOC

Relation between LOC and FPs

Language	LOC/FP
assembly	320
C	128
Cobol	105
Fortan	105
Pascal	90
Ada	70
OO languages	30
4GL languages	20

COCOMO

- CO (Constructive) CO (Cost) MO (Model)
- First version: 1981 by Dr. Barry Boehm
  - Now known now as “COCOMO 81”
- Second version: ADA Cocomo (ADA 87); parameterized exponent reflecting more modern practices and their economies of scale.
- Current Version: Cocomo II (circa 2000)
- Commercial take-offs
  - Costar (Softstarsystems.com)
  - Cost Xpert (CostXpert.com)
- Regression formula, with data taken from historical projects and current project characteristics.

Department of Computer Science and Engineering,
Anna University, Chennai – 600 025.

- The most important factors contributing to a project's duration and cost is the Development Mode
  - **Organic Mode:** The project is developed in a familiar, stable environment, and the product is similar to previously developed products. The product is relatively small, and requires little innovation.
  - **Semidetached Mode:** The project's characteristics are intermediate between Organic and Embedded.
  - **Embedded Mode:** The project is characterized by tight, inflexible constraints and interface requirements. An embedded mode project will require a great deal of innovation.

Feature	Organic	Semidetached	Embedded
Organizational understanding of product and objectives	Thorough	Considerable	General
Experience in working with related software systems	Extensive	Considerable	Moderate
Need for software conformance with pre-established requirements	Basic	Considerable	Full
Need for software conformance with external interface specifications	Basic	Considerable	Full

Feature	Organic	Semidetached	Embedded
Concurrent development of associated new hardware and operational procedures	Some	Moderate	Extensive
Need for innovative data processing architectures, algorithms	Minimal	Some	Considerable
Premium on early completion	Low	Medium	High
Product size range	<50 KDSI	<300KDSI	All

The **Basic COCOMO model** computes effort as a function of program size. The Basic COCOMO equation is:

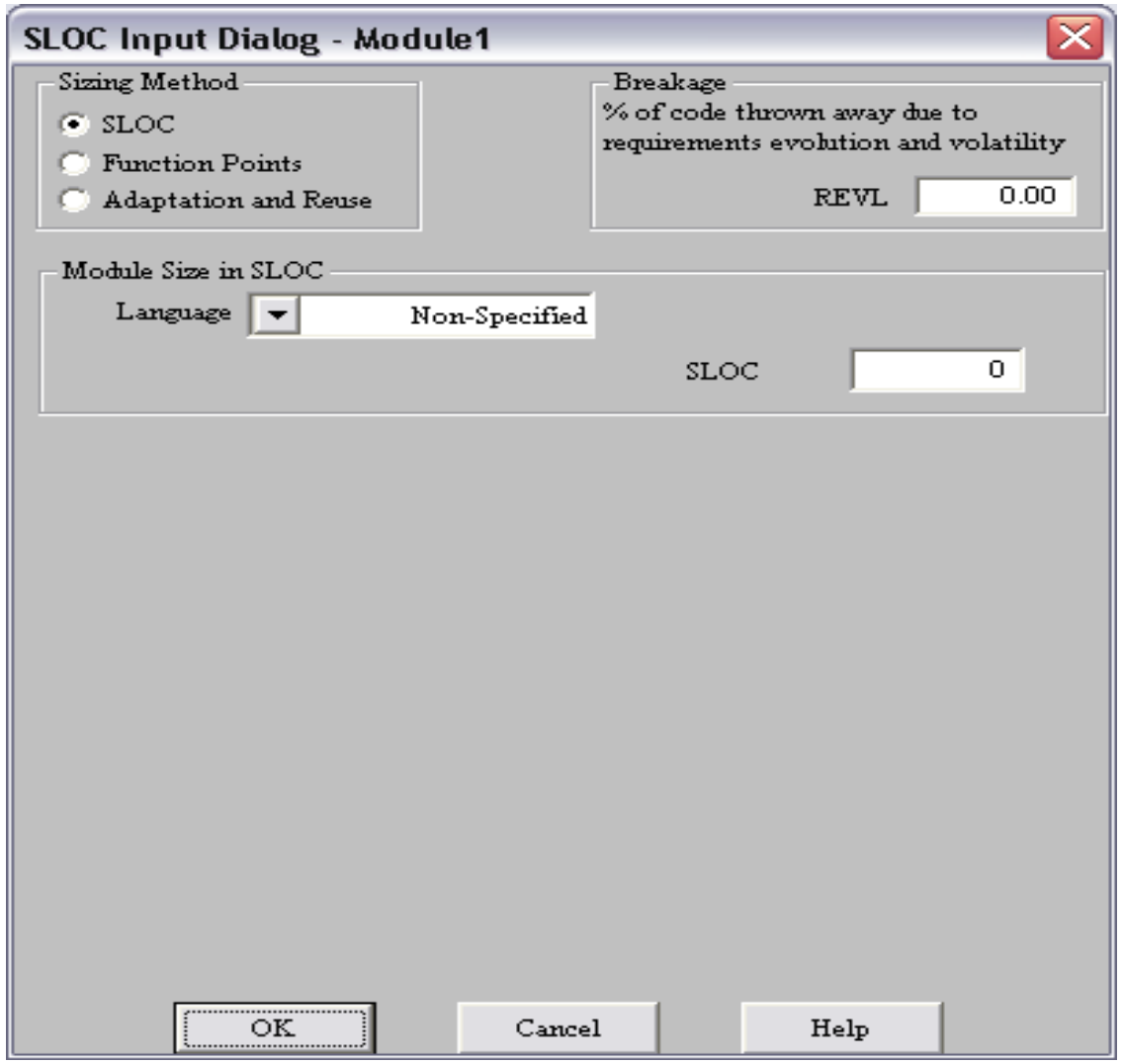
E = aKLOC^b

Effort for three modes of Basic COCOMO

Mode	a	b
Organic	2.4	1.05
Semi-detached	3.0	1.12
Embedded	3.6	1.20

SLOC

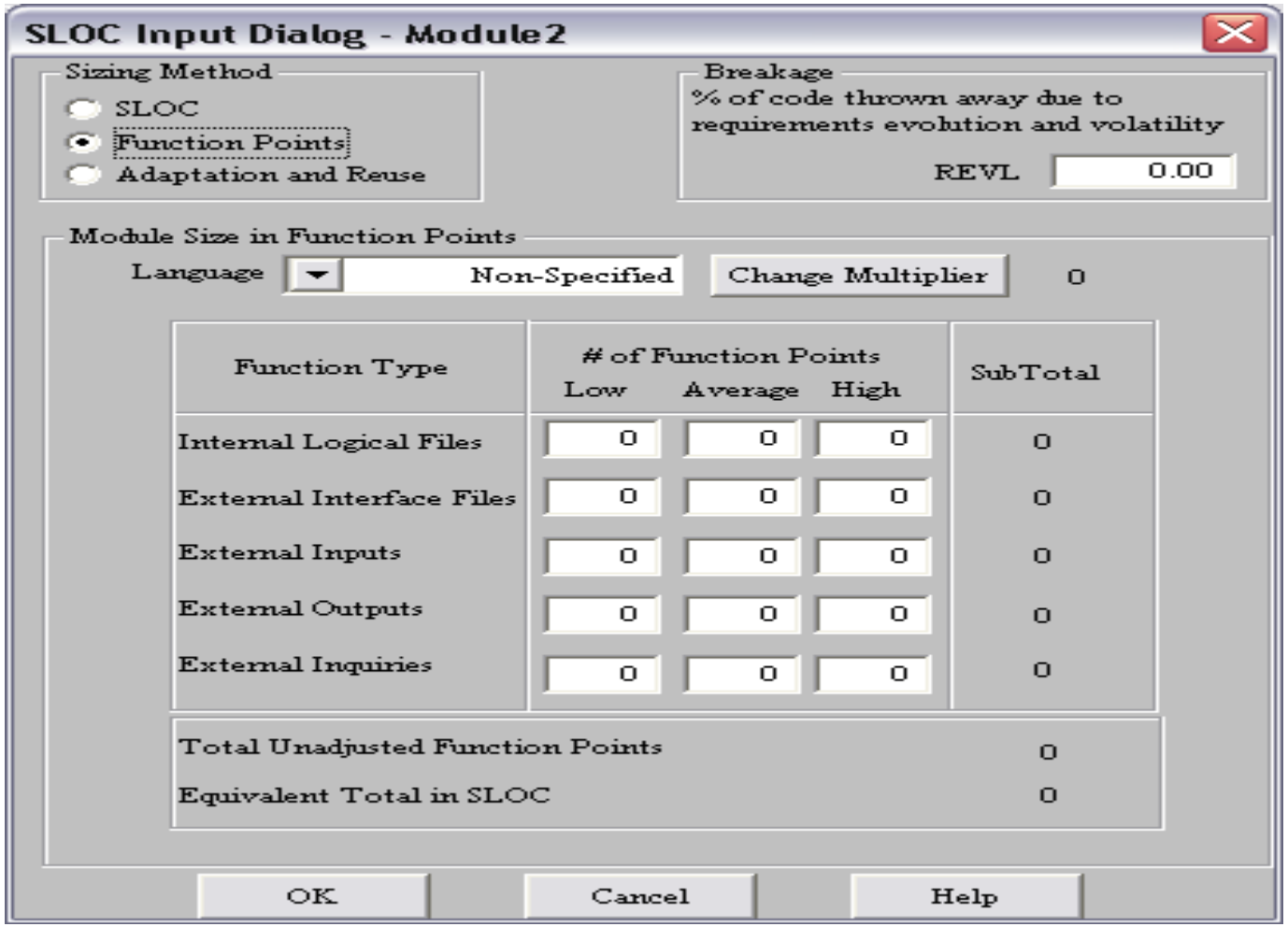
Department of Computer Science and Engineering,
Anna University, Chennai – 600 025.



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.

FUNCTION POINTS

Department of Computer Science and Engineering,
Anna University, Chennai – 600 025.



Department of Computer Science and Engineering,
Anna University, Chennai – 600 025.

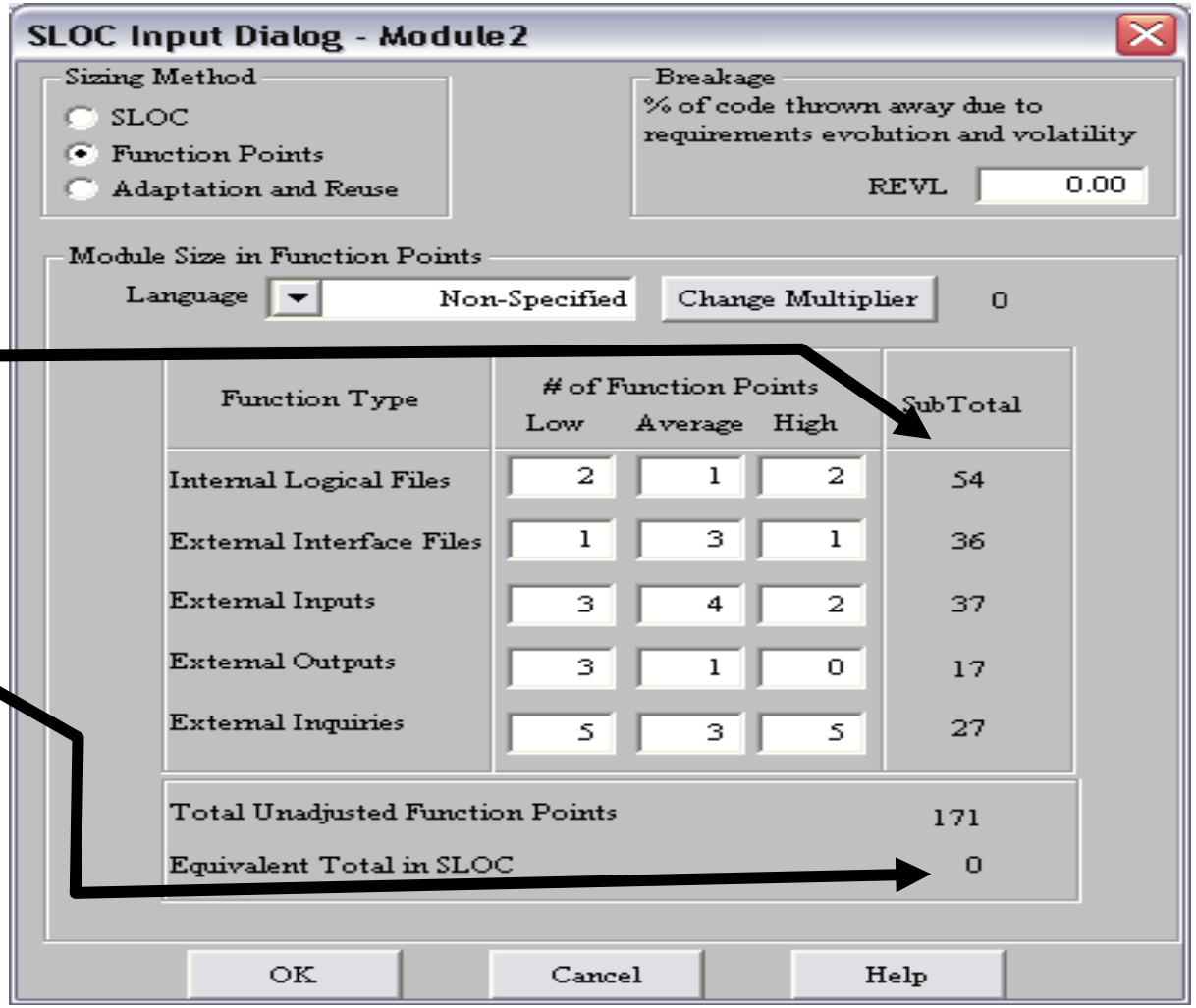
External Input (Inputs)	Count each unique user data or user control input type that (i) enters the external boundary of the software system being measured and (ii) adds or changes data in a logical internal file.
External Output (Outputs)	Count each unique user data or control output type that leaves 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.

FUNCTION POINTS

Department of Computer Science and Engineering,
Anna University, Chennai – 600 025.

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.



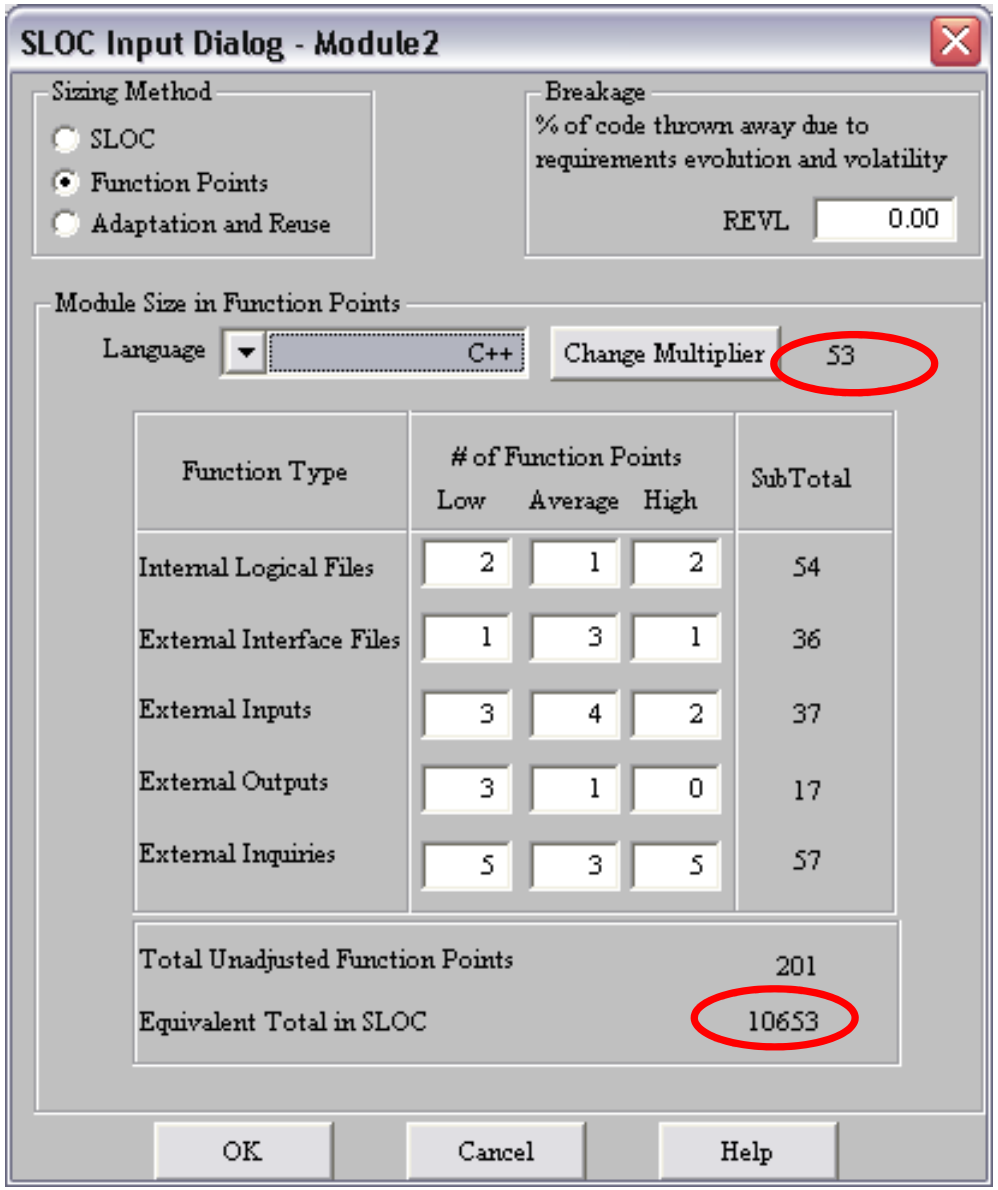
SLOC FOR PROGRAMMING LANGUAGES

Department of Computer Science and Engineering,
Anna University, Chennai – 600 025.

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! ☺



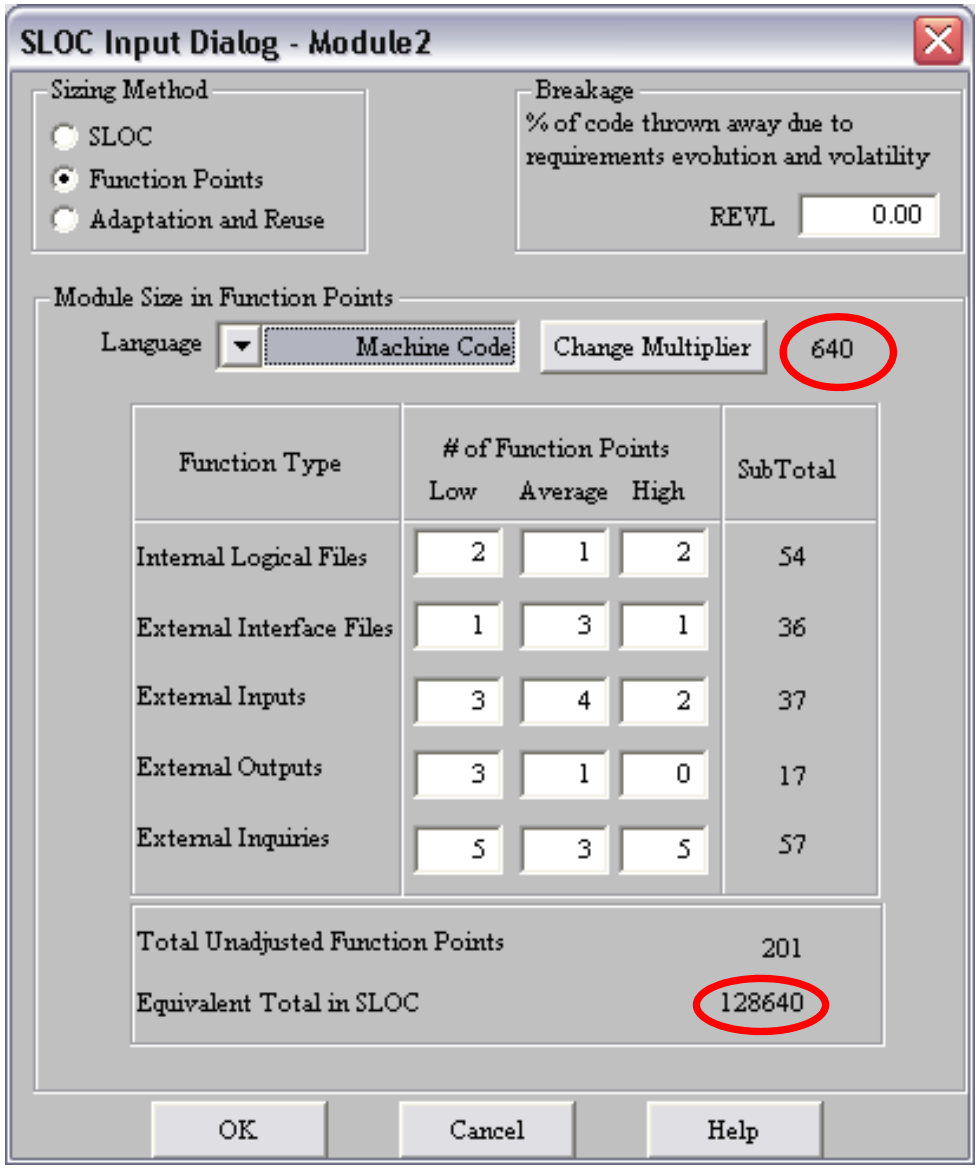
MULTIPLIER

Department of Computer Science and Engineering,
Anna University, Chennai – 600 025.

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 5'th Generation.

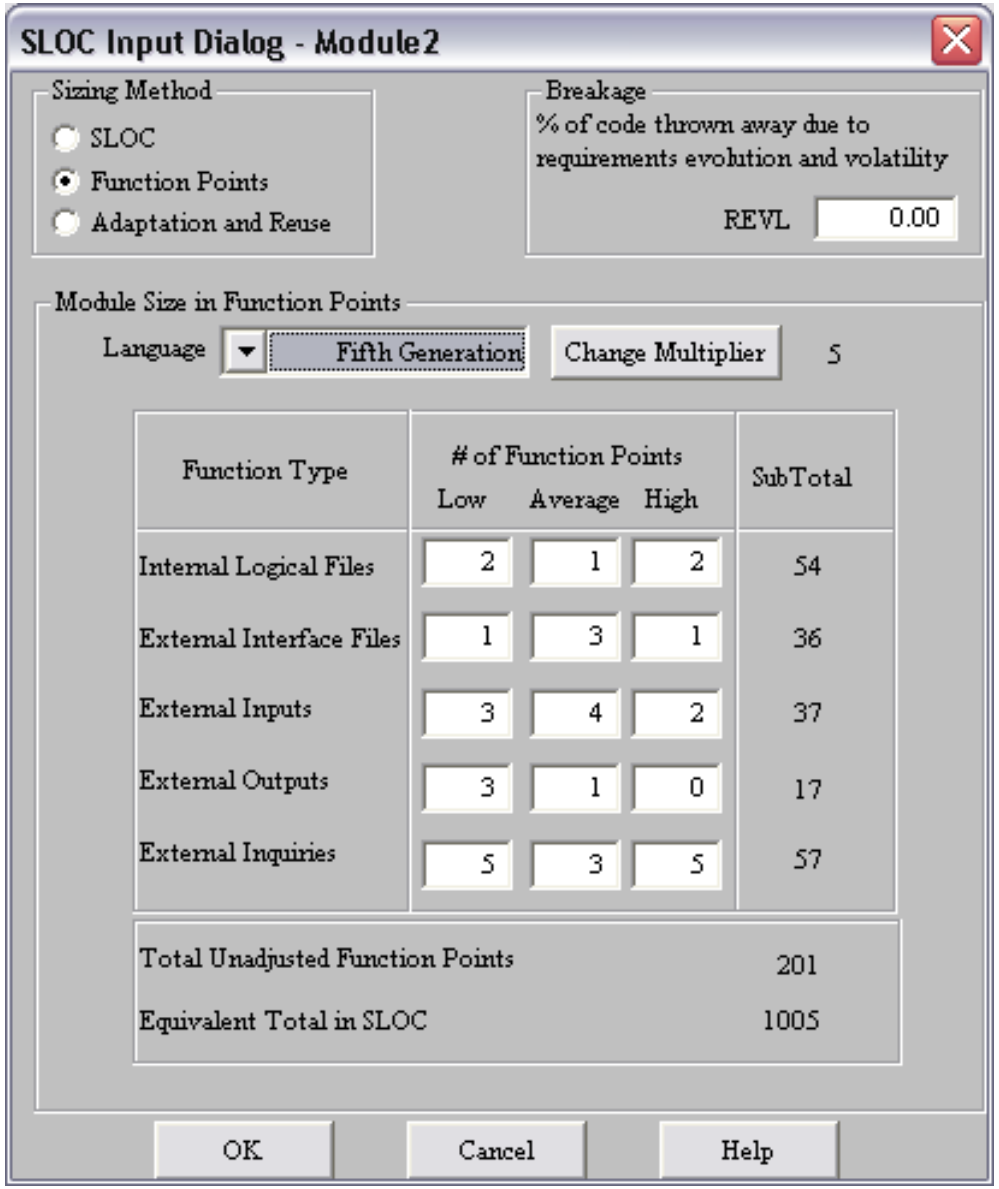


SLOC – 5TH GENERATION LANGUAGES

Department of Computer Science and Engineering,
Anna University, Chennai – 600 025.

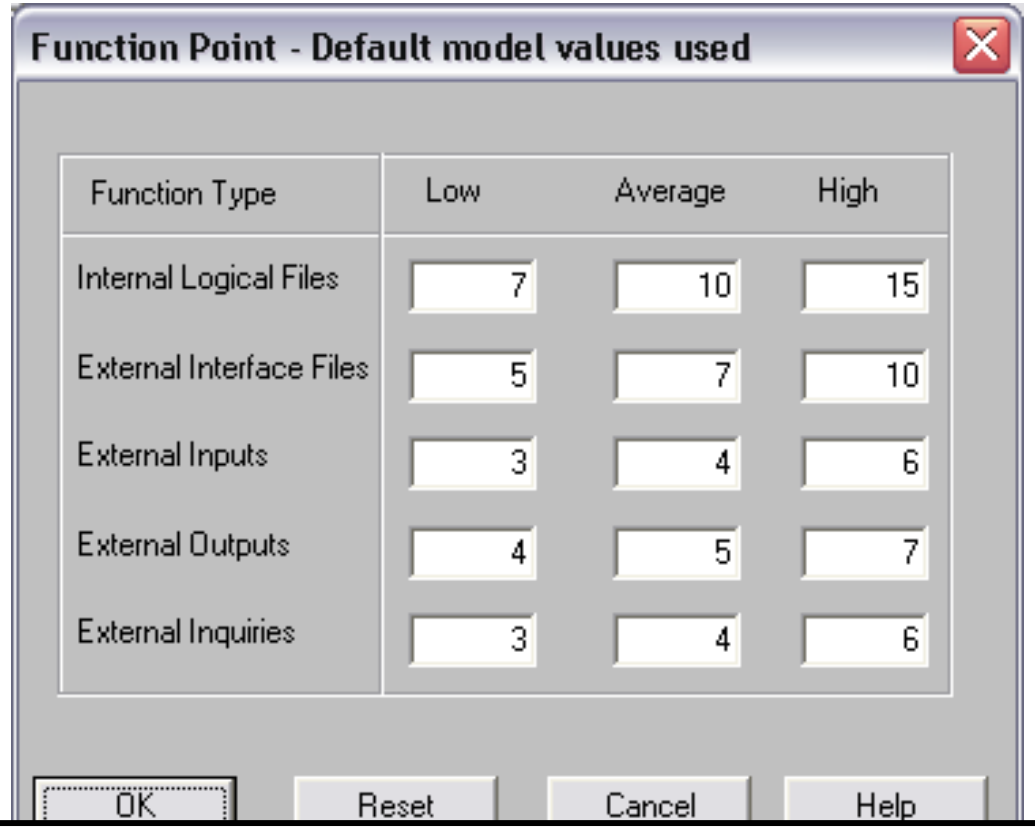
So using a 5'th generation level language would cut our code base by a factor of 285 times according to Cocomo II's default estimation (not calibrated for your environment, not taking into account other factors).

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



FP – DEFAULT MODEL VALUES

Department of Computer Science and Engineering,
Anna University, Chennai – 600 025.



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 Multiplier…

DEFAULT MULTIPLIER VALUES PER LANGUAGE

Department of Computer Science and Engineering,
Anna University, Chennai – 600 025.

Language	Value
Machine Code	640
Assembly, Basic	320
First Generation	320
Assembly, Macro	213
C	128
Fortran77	107
Second Generation	107
Procedural	105
Cobol 85, ANSI	91
High Level	91
Pascal	91
Modula 2	80
Report Generator	80
Third Generation	80
Ada 83	71
Fortran 95	71
Basic, ANSI	64
Lisp	64
Prolog	64
C++	53
Java	53
Ada 95	49
AI Shell	49
Basic, Compiled	49
Forth	49

Language	Value
Simulation Default	46
Database Default	40
Access	38
Visual C++	34
APL	32
Basic, Interpreted	32
Object Oriented	29
Visual Basic 5.0	29
Perl	21
UNIX Shell	21
Fourth Generation	20
PowerBuilder	16
HTML 3.0	15
Query Default	13
Spreadsheet Default	6
Fifth Generation	5
USR_1	1
USR_2	1
USR_3	1
USR_4	1
USR_5	1

THANKS!

BASKARAN RAMACHANDRAN
baaski@annauniv.edu