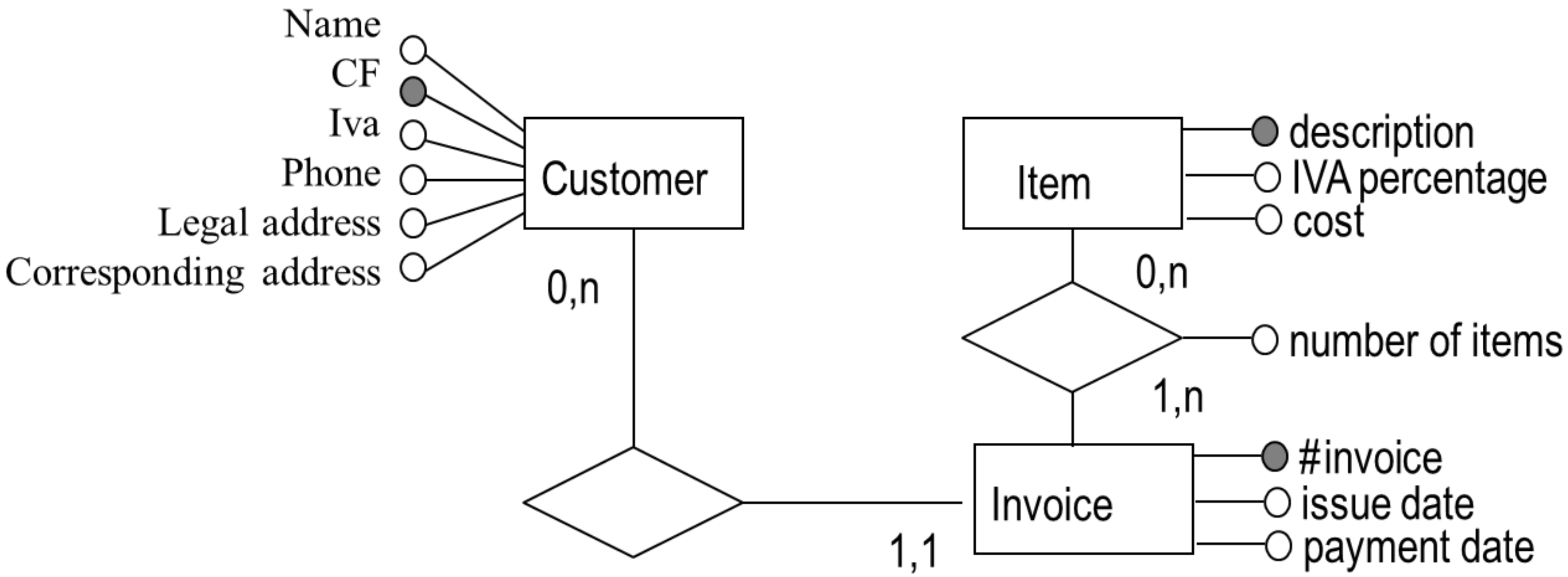
SW Engineering

COCOMO II exercise

The invoice database



Detected functions

- 2 ILF, low complexity
- 1 EIF , low complexity

- 2 EI , low complexity
- 4 EI , high complexity

- 1 EO , medium complexity
- 1 EQ , low complexity

Summary

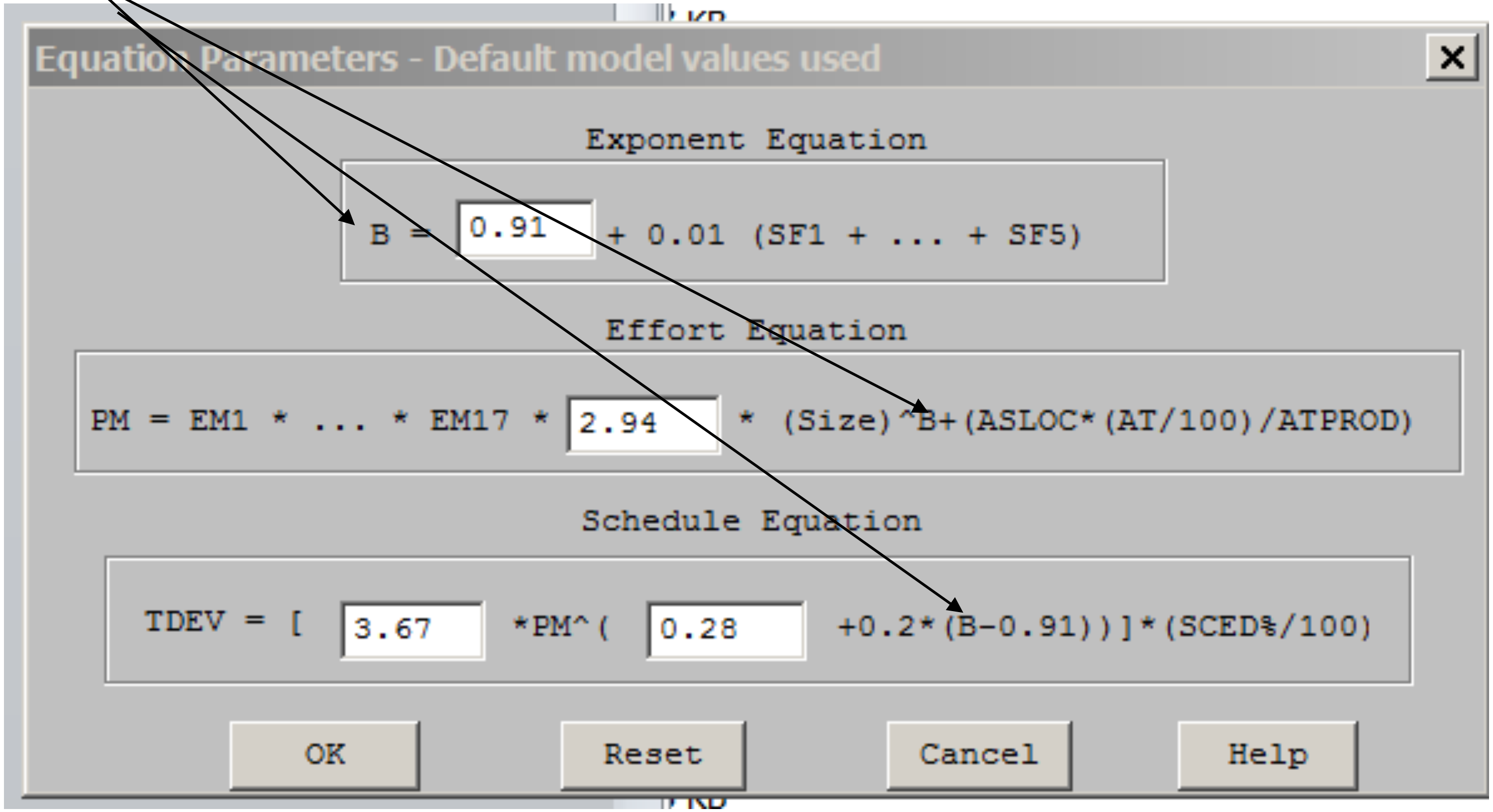
- ILF ELF -> 19
- EI -> 24
- EO -> 5
- EQ -> 3
- TOTAL 51 UFP

Development characteristics

- The application is developed by a group of engineers that graduated this year
- The application is developed in C (!) in a Cobol environment
- There are no particular constraints on the delivery date
- Reusable software does not exist
- We estimate delivery time and cost with the Early Design model

Step 0 : adjust formula parameters

According to the manual it should be E



Equation Parameters - Default model values used

Exponent Equation

B = 0.91 + 0.01 (SF1 + ... + SF5)

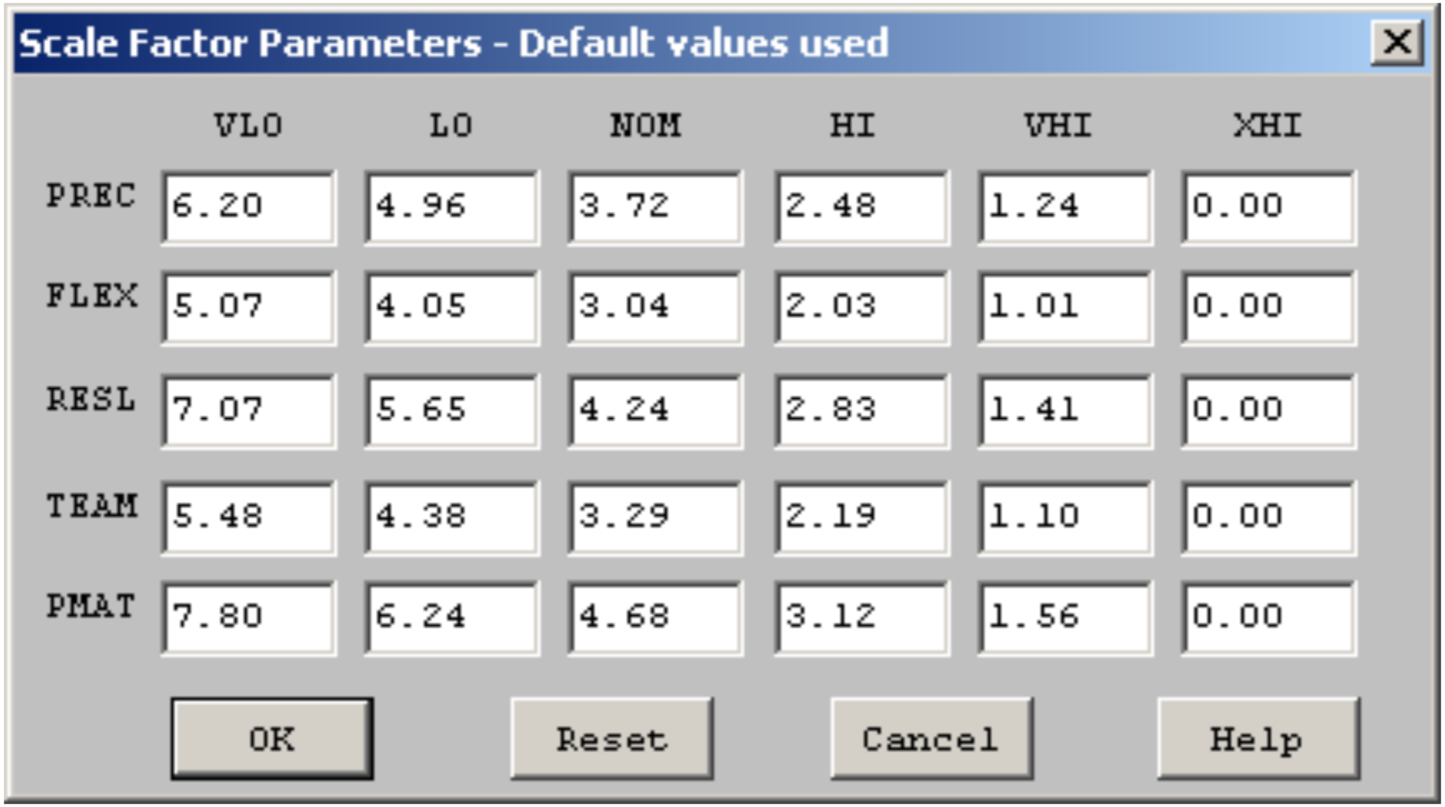
Effort Equation

PM = EM1 * ... * EM17 * 2.94 * (Size)^B+(ASLOC*(AT/100)/ATPROD)

Schedule Equation

TDEV = [3.67 *PM^(0.28 +0.2*(B-0.91))]*(SCED%/100)

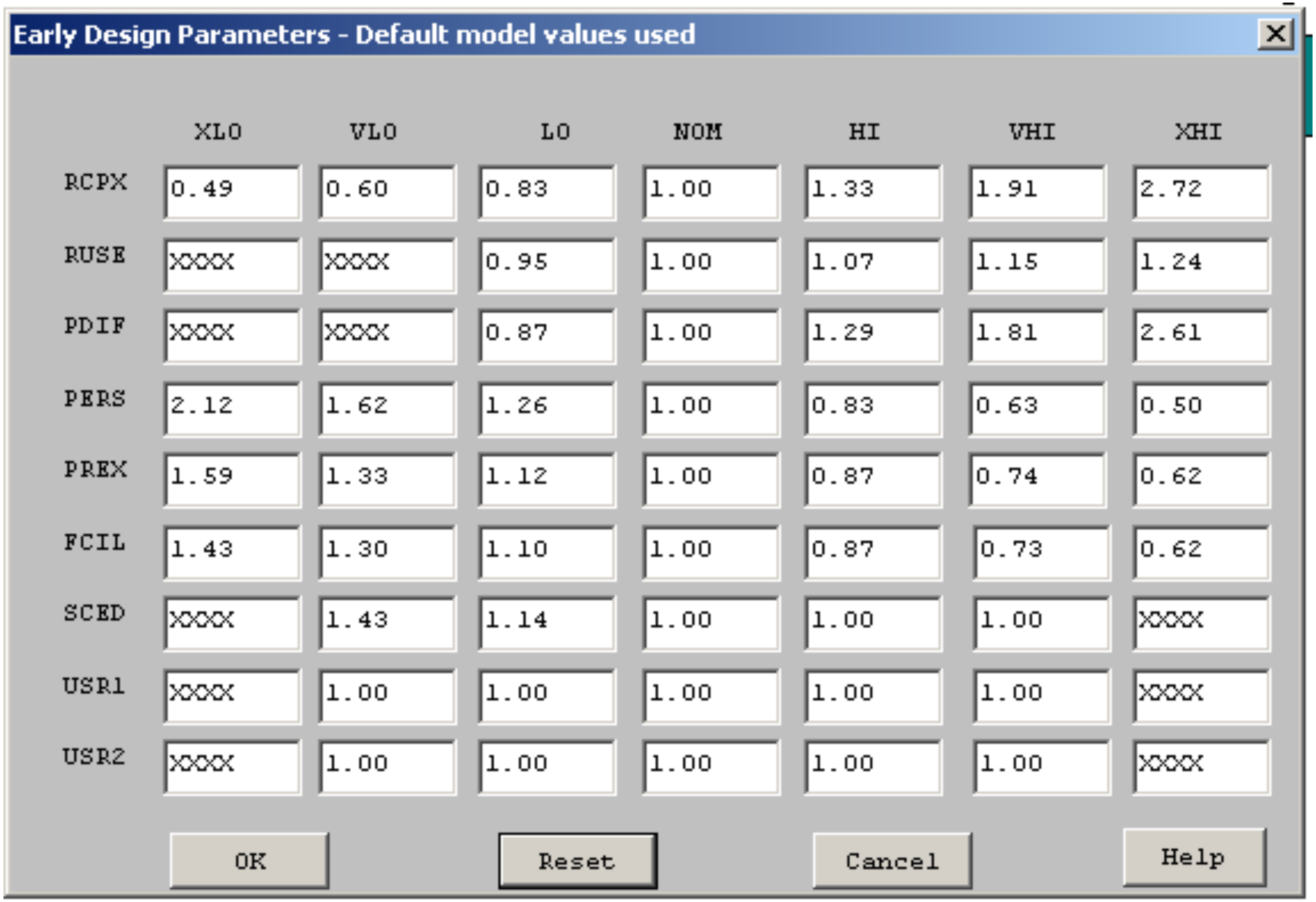
Step 0 : Scale factor



Scale Factor Parameters - Default values used

	VLO	LO	NOM	HI	VHI	XHI
PREC	6.20	4.96	3.72	2.48	1.24	0.00
FLEX	5.07	4.05	3.04	2.03	1.01	0.00
RESL	7.07	5.65	4.24	2.83	1.41	0.00
TEAM	5.48	4.38	3.29	2.19	1.10	0.00
PMAT	7.80	6.24	4.68	3.12	1.56	0.00

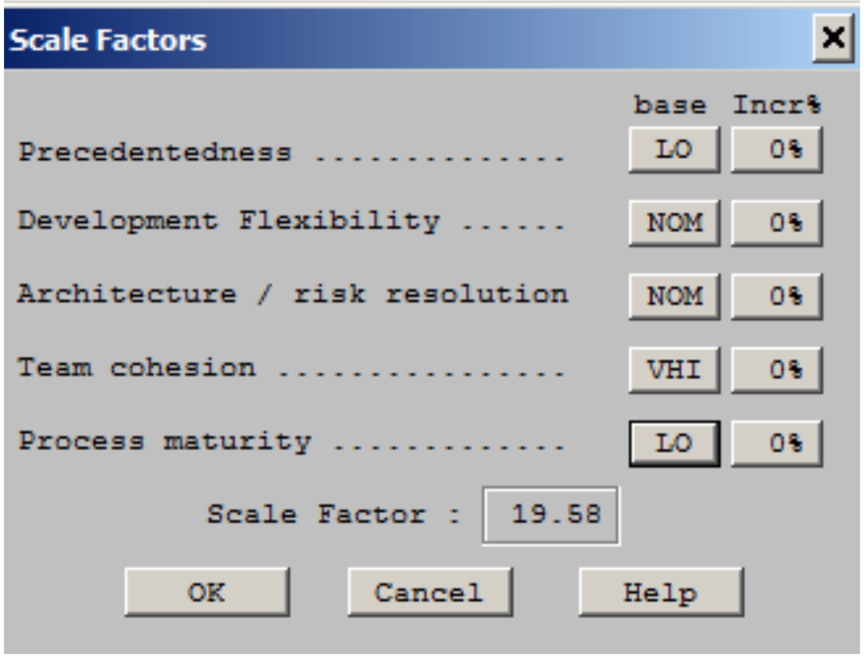
7 Early design multipliers



Early Design Parameters - Default model values used

	XLO	VLO	LO	NOM	HI	VHI	XHI
RCPX	0.49	0.60	0.83	1.00	1.33	1.91	2.72
RUSE	XXXX	XXXX	0.95	1.00	1.07	1.15	1.24
PDIF	XXXX	XXXX	0.87	1.00	1.29	1.81	2.61
PERS	2.12	1.62	1.26	1.00	0.83	0.63	0.50
PREX	1.59	1.33	1.12	1.00	0.87	0.74	0.62
FCIL	1.43	1.30	1.10	1.00	0.87	0.73	0.62
SCED	XXXX	1.43	1.14	1.00	1.00	1.00	XXXX
USR1	XXXX	1.00	1.00	1.00	1.00	1.00	XXXX
USR2	XXXX	1.00	1.00	1.00	1.00	1.00	XXXX

Step 1 : E calculation



Scale Factors

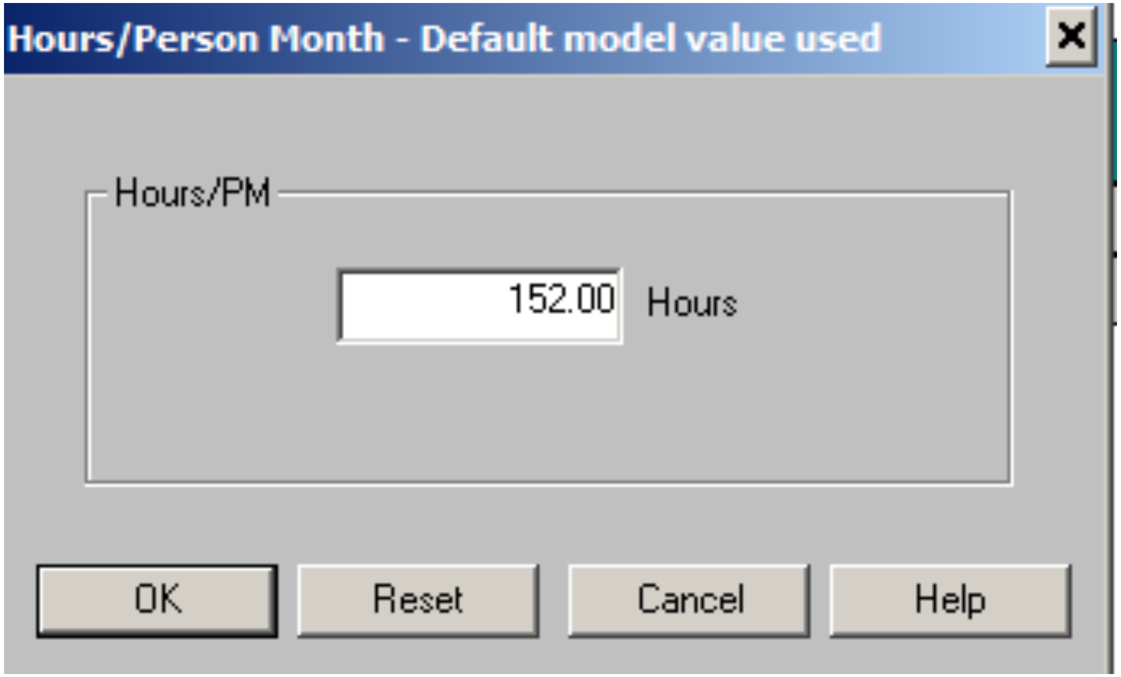
	base	Incr%
Precedentedness	LO	0%
Development Flexibility	NOM	0%
Architecture / risk resolution	NOM	0%
Team cohesion	VHI	0%
Process maturity	LO	0%

Scale Factor : 19.58

- Previous experience LOW
- Flexibility NOM
- RES NOM
- Team cohesion VERY HIGH
- Process maturity LOW

$$E = 0.91 + 0.01 \sum_{i=1}^{5} SF_i = 0.91 + 0.1*19.58 = 1.1058$$

Step 1 : Working hours

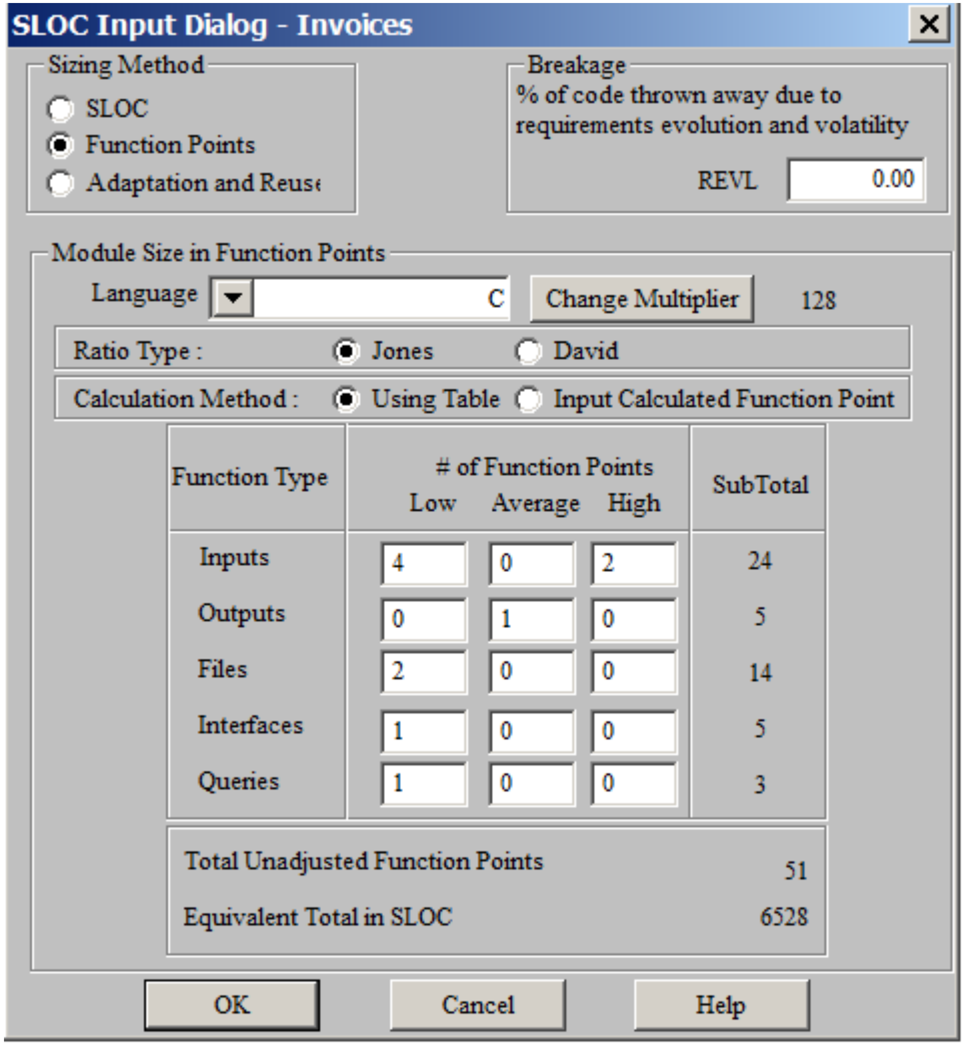


Hours/Person Month - Default model value used

Hours/PM

152.00 Hours

Step 2: new module (FP driven)



Sizing Method

○ SLOC
● Function Points
○ Adaptation and Reuse

Breakage

% of code thrown away due to requirements evolution and volatility

REVL 0.00

Module Size in Function Points

Language C

Change Multiplier 128

Ratio Type : ● Jones ○ David

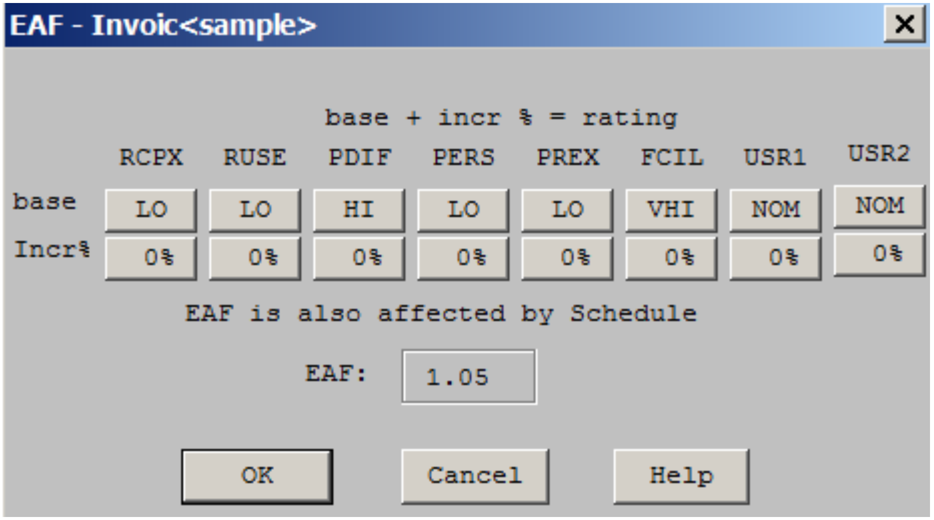
Calculation Method : ● Using Table ○ Input Calculated Function Point

Function Type	# of Function Points Low	 Average	 High	SubTotal
Inputs	4	0	2	24
Outputs	0	1	0	5
Files	2	0	0	14
Interfaces	1	0	0	5
Queries	1	0	0	3

Total Unadjusted Function Points 51

Equivalent Total in SLOC 6528

Step2 : Effort multipliers

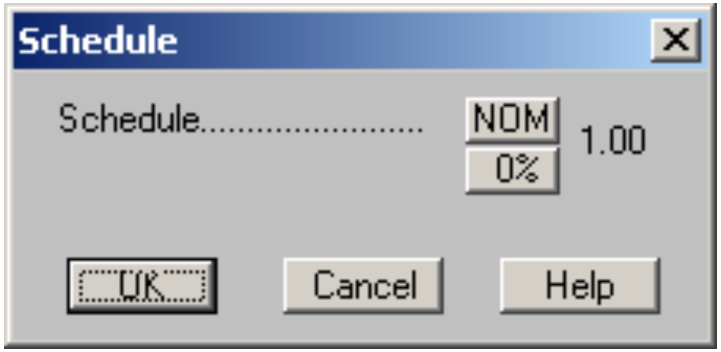

EAF - Invoic<sample>

base + incr % = rating

	RCPX	RUSE	PDIF	PERS	PREX	FCIL	USR1	USR2
base	LO	LO	HI	LO	LO	VHI	NOM	NOM
Incr%	0%	0%	0%	0%	0%	0%	0%	0%

EAF is also affected by Schedule

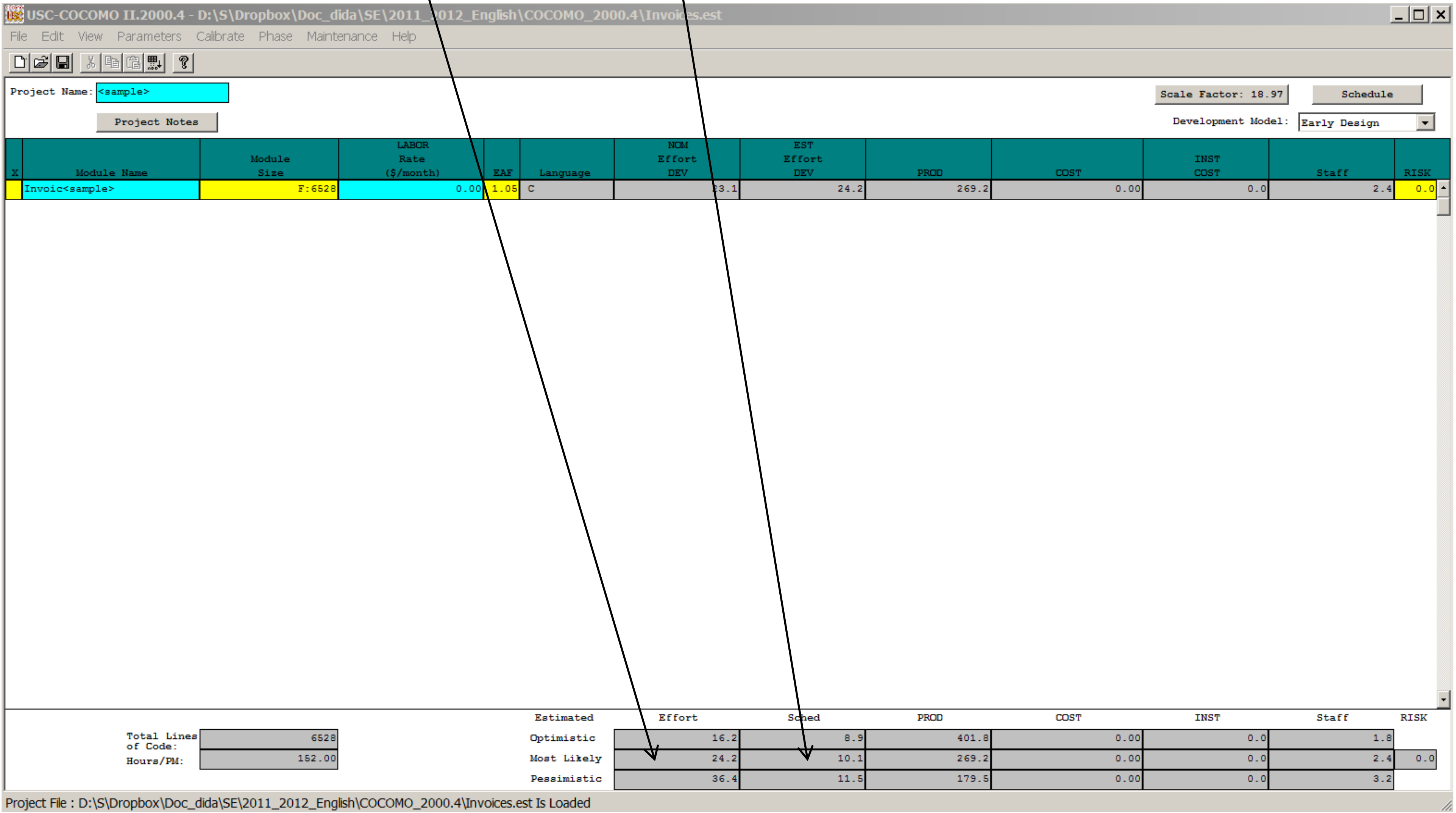
EAF: 1.05


Schedule

Schedule........................ NOM 0% 1.00

- Complexity LOW
- Reuse LOW
- Platform diff. HIGH
- Personnel LOW
- Experience LOW
- Facilities VHI
- Schedule NOM

M and T estimation


USC-COCOMO II.2000.4 - D:\S\Dropbox\Doc_dida\SE\2011_2012_English\COCOMO_2000.4\Invoices.est
File Edit View Parameters Calibrate Phase Maintenance Help
Project Notes
Schedule

Project Name: <sample>

Scale Factor: 18.97

Development Model: Early Design

X	Module Name	Module Size	LABOR Rate ($/month)	EAF	Language	NOM Effort DEV	EST Effort DEV	PROD	COST	INST COST	Staff	RISK
	Invoic<sample>	F:6528	0.00	1.05	C	23.1	24.2	269.2	0.00	0.0	2.4	0.0

Total Lines of Code: 6528

Hours/PM: 152.00

Estimated	Effort	Sched	PROD	COST	INST	Staff	RISK
Optimistic	16.2	8.9	401.8	0.00	0.0	1.8	
Most Likely	24.2	10.1	269.2	0.00	0.0	2.4	0.0
Pessimistic	36.4	11.5	179.5	0.00	0.0	3.2	

Project File : D:\S\Dropbox\Doc_dida\SE\2011_2012_English\COCOMO_2000.4\Invoices.est Is Loaded

Planning

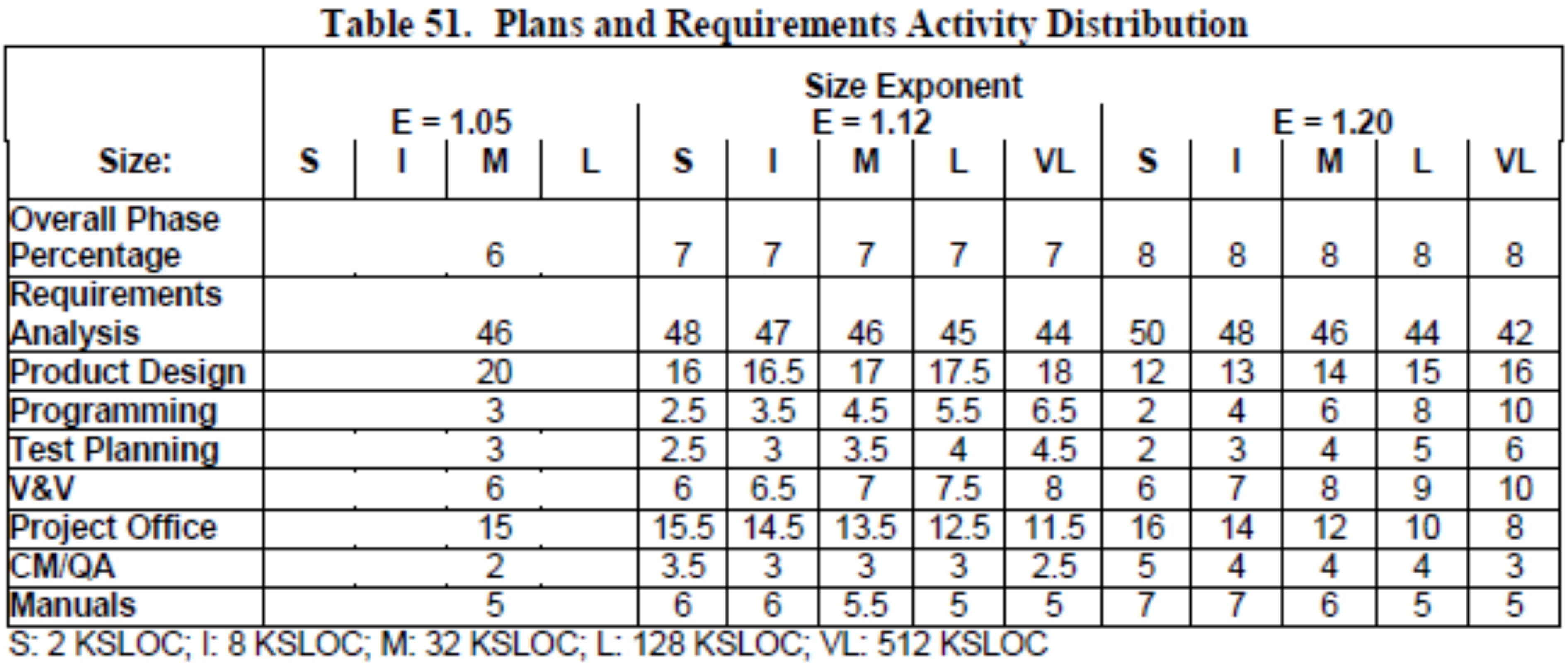
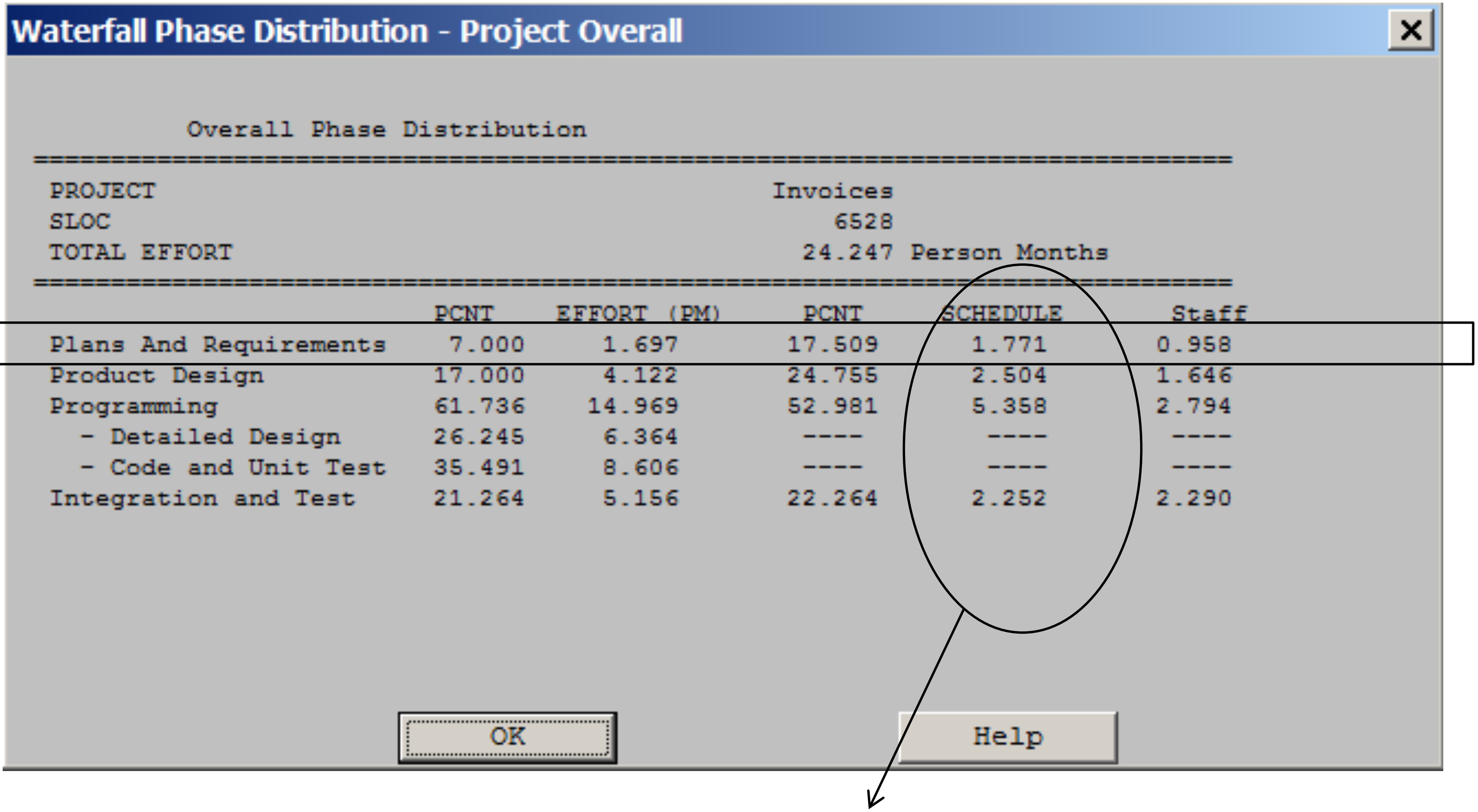


Table 51. Plans and Requirements Activity Distribution

	Size Exponent													
	E = 1.05				E = 1.12					E = 1.20				
Size:	S	I	M	L	S	I	M	L	VL	S	I	M	L	VL
Overall Phase Percentage			6		7	7	7	7	7	8	8	8	8	8
Requirements Analysis			46		48	47	46	45	44	50	48	46	44	42
Product Design			20		16	16.5	17	17.5	18	12	13	14	15	16
Programming			3		2.5	3.5	4.5	5.5	6.5	2	4	6	8	10
Test Planning			3		2.5	3	3.5	4	4.5	2	3	4	5	6
V&V			6		6	6.5	7	7.5	8	6	7	8	9	10
Project Office			15		15.5	14.5	13.5	12.5	11.5	16	14	12	10	8
CM/QA			2		3.5	3	3	3	2.5	5	4	4	4	3
Manuals			5		6	6	5.5	5	5	7	7	6	5	5

S: 2 KSLOC; I: 8 KSLOC; M: 32 KSLOC; L: 128 KSLOC; VL: 512 KSLOC

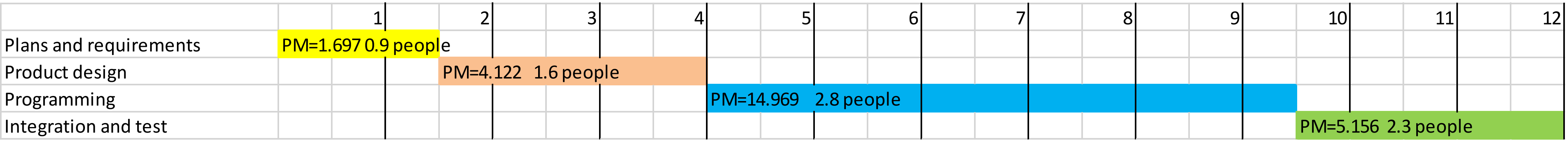
GANTT



Waterfall Phase Distribution - Project Overall

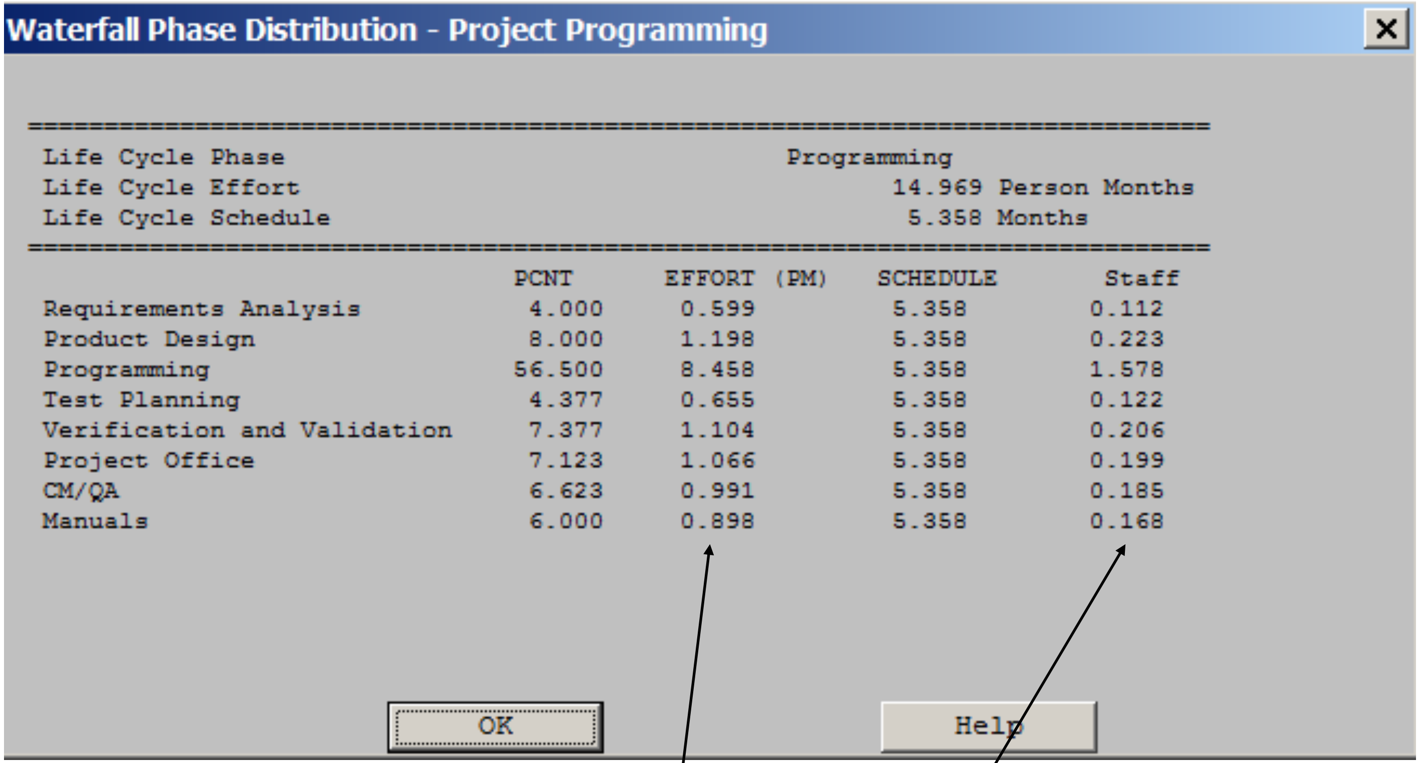
```
         Overall Phase Distribution
==============================================================================
 PROJECT                                                Invoices
 SLOC                                                       6528
 TOTAL EFFORT                                             24.247 Person Months
==============================================================================
                         PCNT     EFFORT (PM)      PCNT     SCHEDULE      Staff
 Plans And Requirements   7.000      1.697       17.509      1.771        0.958
 Product Design          17.000      4.122       24.755      2.504        1.646
 Programming             61.736     14.969       52.981      5.358        2.794
   - Detailed Design     26.245      6.364        ----        ----         ----
   - Code and Unit Test  35.491      8.606        ----        ----         ----
 Integration and Test    21.264      5.156       22.264      2.252        2.290
```

Out of estimation



	1	2	3	4	5	6	7	8	9	10	11	12
Plans and requirements	PM=1.697 0.9 people											
Product design		PM=4.122 1.6 people										
Programming					PM=14.969 2.8 people							
Integration and test										PM=5.156 2.3 people		

Effort distribution

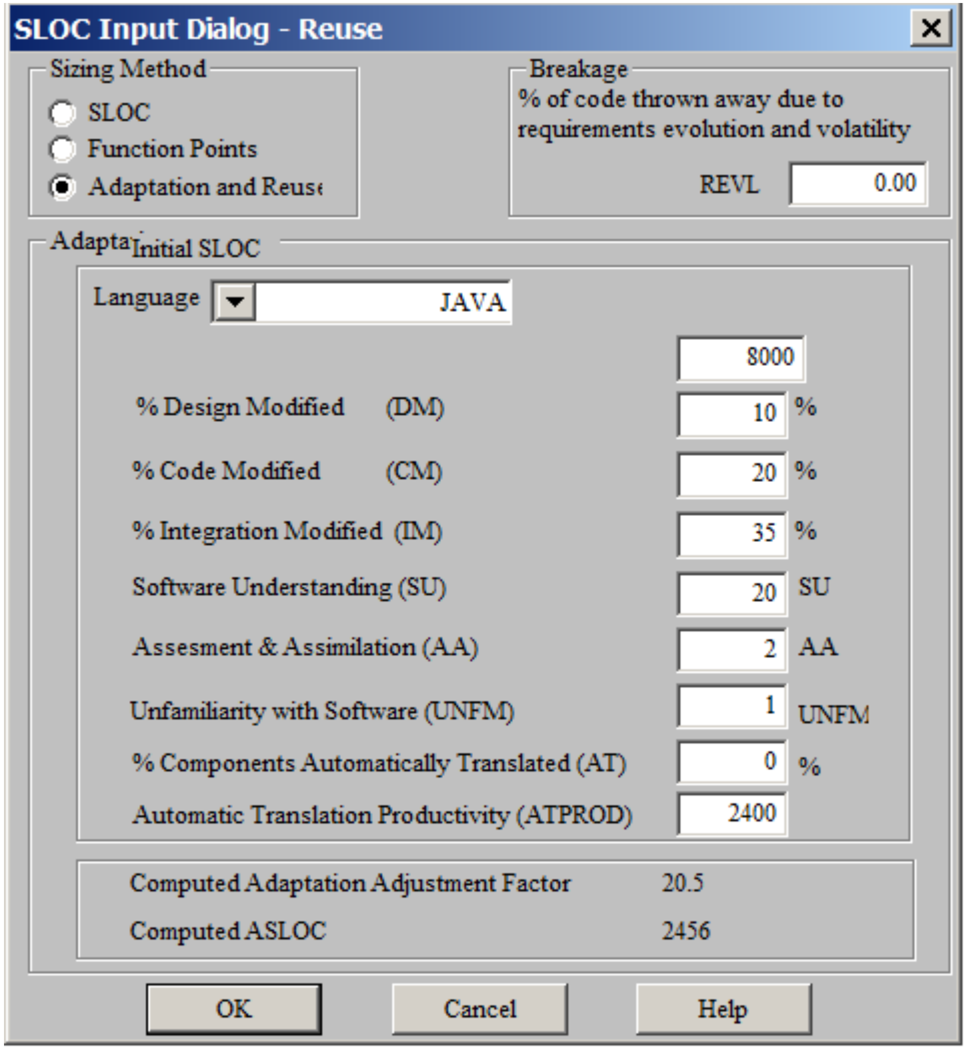


Waterfall Phase Distribution - Project Programming

```
=============================================================================
 Life Cycle Phase                                Programming
 Life Cycle Effort                                      14.969 Person Months
 Life Cycle Schedule                                     5.358 Months
=============================================================================
                               PCNT      EFFORT (PM)    SCHEDULE       Staff
 Requirements Analysis          4.000      0.599          5.358         0.112
 Product Design                 8.000      1.198          5.358         0.223
 Programming                   56.500      8.458          5.358         1.578
 Test Planning                  4.377      0.655          5.358         0.122
 Verification and Validation    7.377      1.104          5.358         0.206
 Project Office                 7.123      1.066          5.358         0.199
 CM/QA                          6.623      0.991          5.358         0.185
 Manuals                        6.000      0.898          5.358         0.168
```

	1	2	3	4	5	6	7	8	9	10	11	12
Plans and requirements	PM=1.784 0.9 people											
Product design		PM=4.169 1.6 people										
Programming					PM=15.142	2.8 people						
Integration and test										PM=5.21 2.3 people		

Reuse example



SLOC Input Dialog - Reuse

Sizing Method

○ SLOC
○ Function Points
● Adaptation and Reus

Breakage
% of code thrown away due to requirements evolution and volatility

REVL 0.00

Adapta Initial SLOC

Language: JAVA

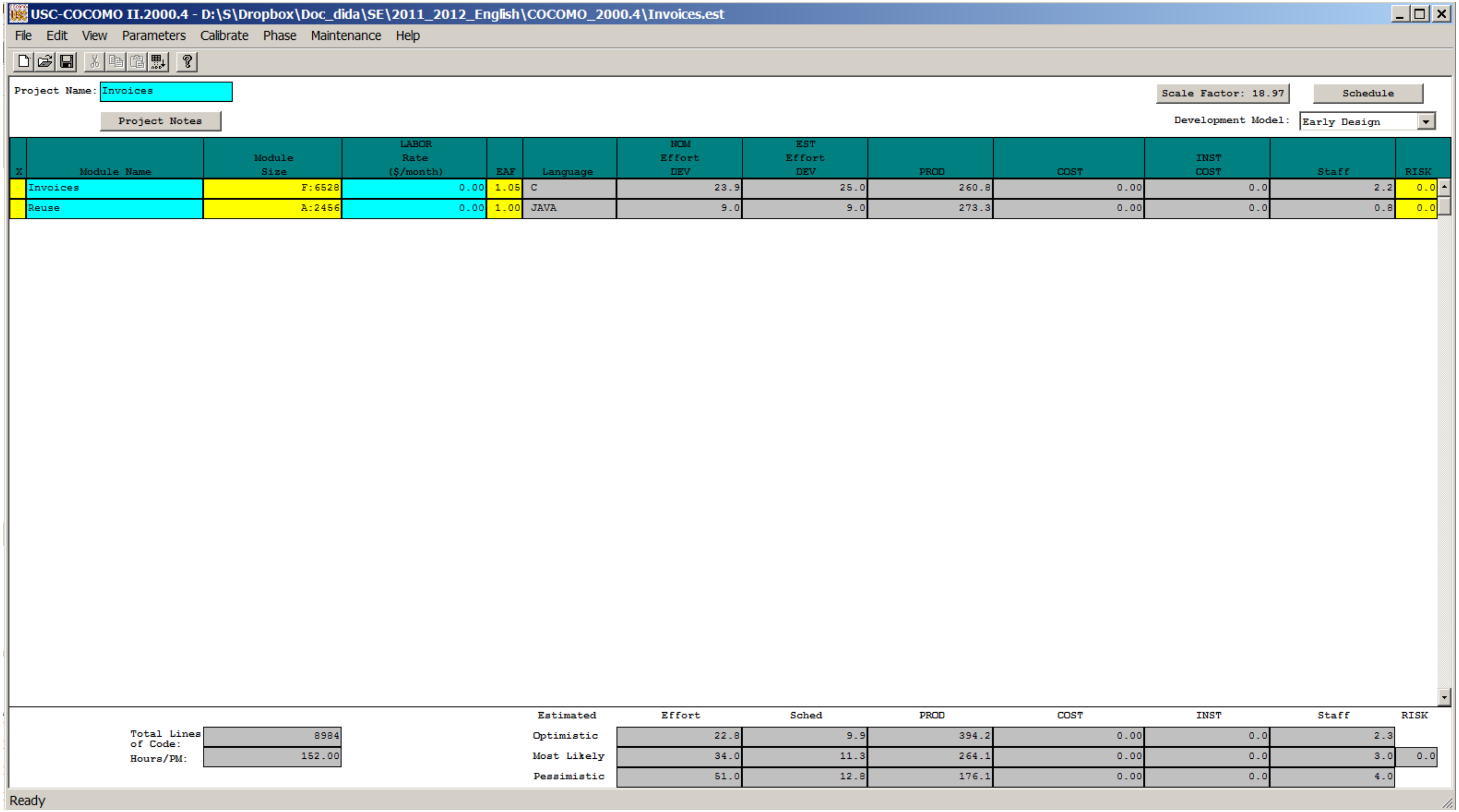
	8000	
% Design Modified (DM)	10	%
% Code Modified (CM)	20	%
% Integration Modified (IM)	35	%
Software Understanding (SU)	20	SU
Assesment & Assimilation (AA)	2	AA
Unfamiliarity with Software (UNFM)	1	UNFM
% Components Automatically Translated (AT)	0	%
Automatic Translation Productivity (ATPROD)	2400	

Computed Adaptation Adjustment Factor 20.5
Computed ASLOC 2456

8kLOC sw module for browsing a table on the Web

- **well written and documented code SU=20**
- **simple assessment and adaptation AA=2**
- **unknown source code UNF=1.0**
- **% of project being modified 10% DM=10**
- **% of code being modified 20% CM=20**
- **% additional integration effort 35% IM=35**

New estimation



USC-COCOMO II.2000.4 - D:\S\Dropbox\Doc_dida\SE\2011_2012_English\COCOMO_2000.4\Invoices.est

Project Name: Invoices

Scale Factor: 18.97

Development Model: Early Design

X	Module Name	Module Size	LABOR Rate ($/month)	EAF	Language	NOM Effort DEV	EST Effort DEV	PROD	COST	INST COST	Staff	RISK
	Invoices	F:6528	0.00	1.05	C	23.9	25.0	260.8	0.00	0.0	2.2	0.0
	Reuse	A:2456	0.00	1.00	JAVA	9.0	9.0	273.3	0.00	0.0	0.8	0.0

Total Lines of Code: 8984

Hours/PM: 152.00

Estimated	Effort	Sched	PROD	COST	INST	Staff	RISK
Optimistic	22.8	9.9	394.2	0.00	0.0	2.3	
Most Likely	34.0	11.3	264.1	0.00	0.0	3.0	0.0
Pessimistic	51.0	12.8	176.1	0.00	0.0	4.0	

Ready

Some note on SLOC

- Code size is expressed in thousands of source lines of code (KSLOC) but the actual software uses SLOC
- Non-delivered support software such as test drivers is not included in the count
- The goal is to measure the amount of intellectual work put into program development.
- Defining a line of code is difficult (different languages, executable statements, data declarations, comments, programming style)
- In COCOMO II, the **logical** source statement has been chosen as the standard line of code
- The Software Engineering Institute (SEI) definition checklist for a logical source statement is used by Cocomo II in defining the line of code measure

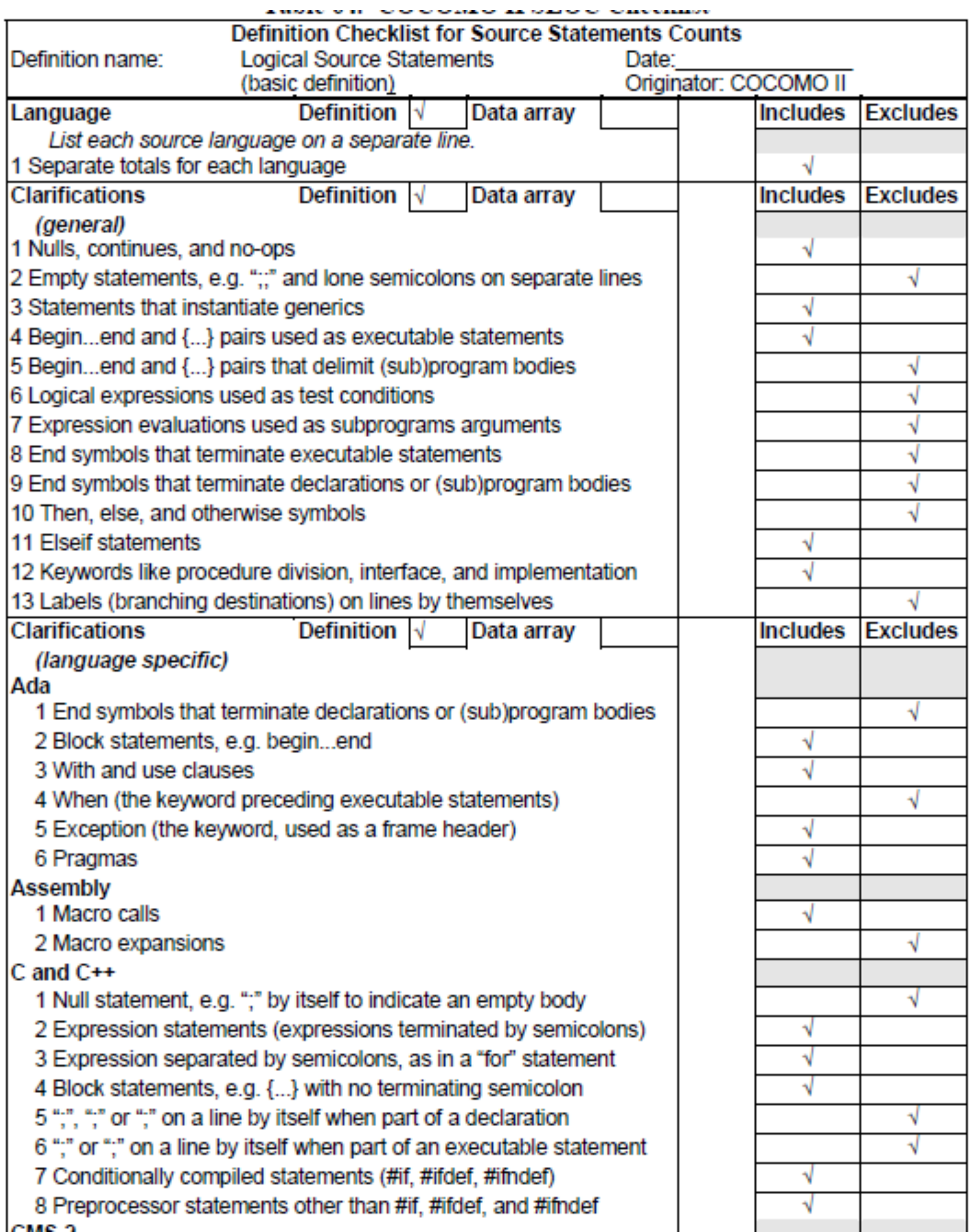
Logical vs. physical

```
if (x > 0) {
printf(“x is a positive number”);
}
```

vs

```
if (x > 0) printf(“x is a positive number”);
```

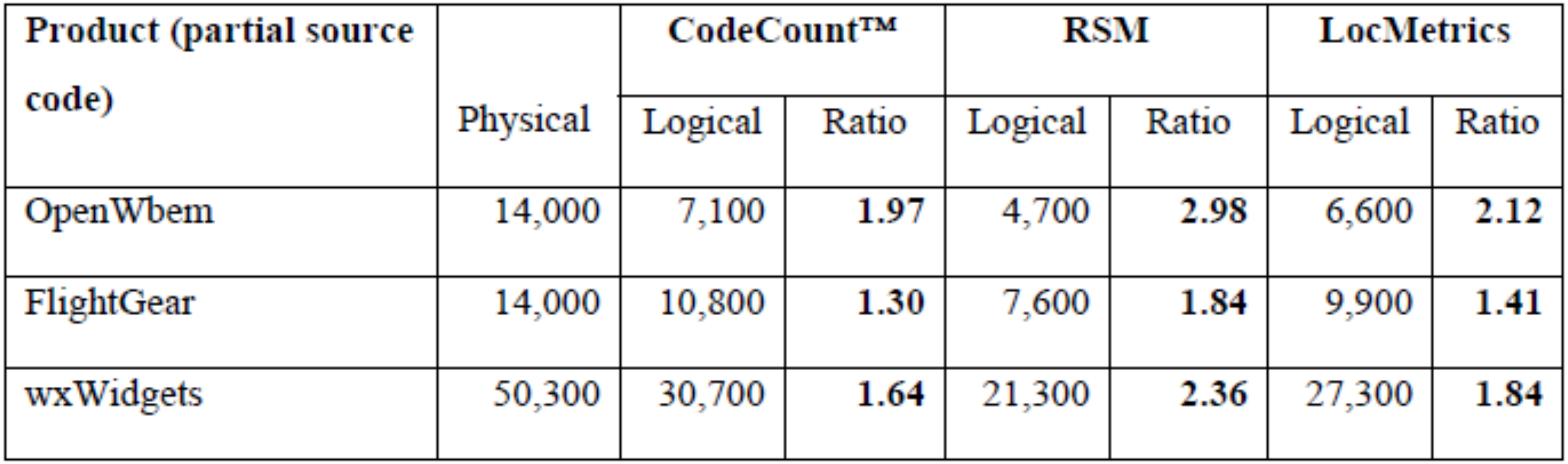
Logical vs. physical



Definition Checklist for Source Statements Counts						
Definition name:	Logical Source Statements (basic definition)			Date:______________ Originator: COCOMO II		
Language	**Definition**	√	**Data array**		**Includes**	**Excludes**
List each source language on a separate line.						
1 Separate totals for each language					√	
Clarifications	**Definition**	√	**Data array**		**Includes**	**Excludes**
(general)						
1 Nulls, continues, and no-ops					√	
2 Empty statements, e.g. “;;” and lone semicolons on separate lines						√
3 Statements that instantiate generics					√	
4 Begin...end and {...} pairs used as executable statements					√	
5 Begin...end and {...} pairs that delimit (sub)program bodies						√
6 Logical expressions used as test conditions						√
7 Expression evaluations used as subprograms arguments						√
8 End symbols that terminate executable statements						√
9 End symbols that terminate declarations or (sub)program bodies						√
10 Then, else, and otherwise symbols						√
11 Elseif statements					√	
12 Keywords like procedure division, interface, and implementation					√	
13 Labels (branching destinations) on lines by themselves						√
Clarifications	**Definition**	√	**Data array**		**Includes**	**Excludes**
(language specific)						
Ada						
1 End symbols that terminate declarations or (sub)program bodies						√
2 Block statements, e.g. begin...end					√	
3 With and use clauses					√	
4 When (the keyword preceding executable statements)						√
5 Exception (the keyword, used as a frame header)					√	
6 Pragmas					√	
Assembly						
1 Macro calls					√	
2 Macro expansions						√
C and C++						
1 Null statement, e.g. “;” by itself to indicate an empty body						√
2 Expression statements (expressions terminated by semicolons)					√	
3 Expression separated by semicolons, as in a “for” statement					√	
4 Block statements, e.g. {...} with no terminating semicolon					√	
5 “;”, “;” or “;” on a line by itself when part of a declaration						√
6 “;” or “;” on a line by itself when part of an executable statement						√
7 Conditionally compiled statements (#if, #ifdef, #ifndef)					√	
8 Preprocessor statements other than #if, #ifdef, and #ifndef					√	
CMS-2						

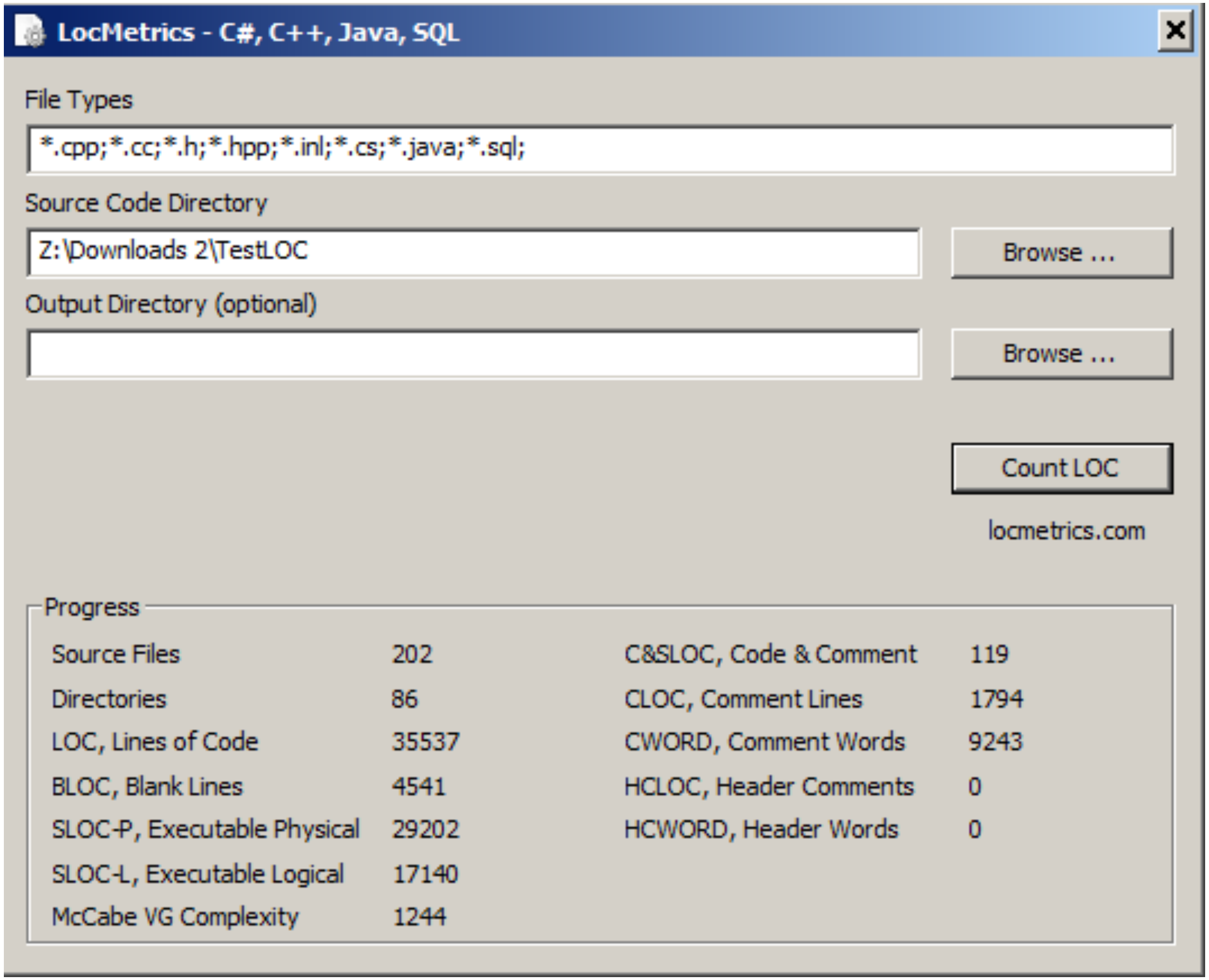
06_B_fpex1.21

Logical vs. physical



Product (partial source code)	Physical	CodeCount™		RSM		LocMetrics	
		Logical	Ratio	Logical	Ratio	Logical	Ratio
OpenWbem	14,000	7,100	**1.97**	4,700	**2.98**	6,600	**2.12**
FlightGear	14,000	10,800	**1.30**	7,600	**1.84**	9,900	**1.41**
wxWidgets	50,300	30,700	**1.64**	21,300	**2.36**	27,300	**1.84**

Logical vs. physical: LocMetrics



File Types

.cpp;.cc;*.h;*.hpp;*.inl;*.cs;*.java;*.sql;

Source Code Directory

Z:\Downloads 2\TestLOC

Output Directory (optional)

Progress

Source Files	202	C&SLOC, Code & Comment	119
Directories	86	CLOC, Comment Lines	1794
LOC, Lines of Code	35537	CWORD, Comment Words	9243
BLOC, Blank Lines	4541	HCLOC, Header Comments	0
SLOC-P, Executable Physical	29202	HCWORD, Header Words	0
SLOC-L, Executable Logical	17140		
McCabe VG Complexity	1244		

Logical vs. physical USC-UCC (http://csse.usc.edu/ucc_wp/)

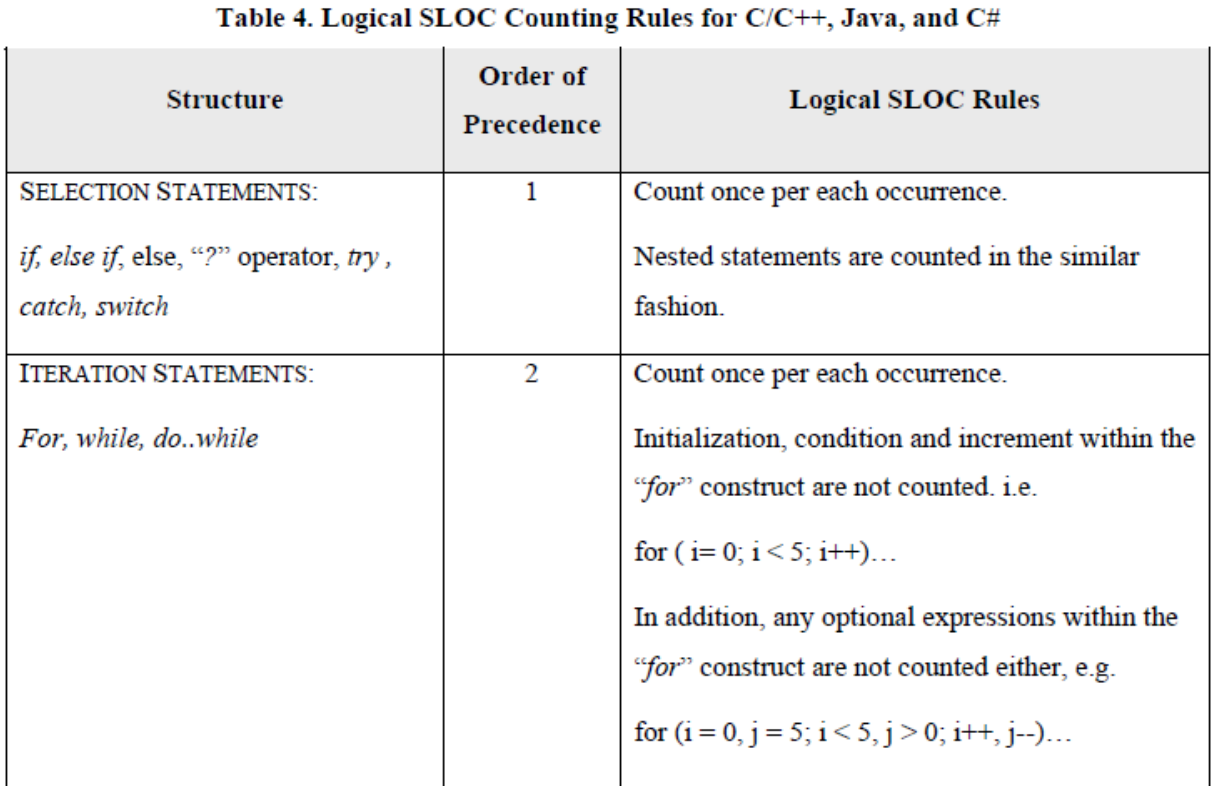


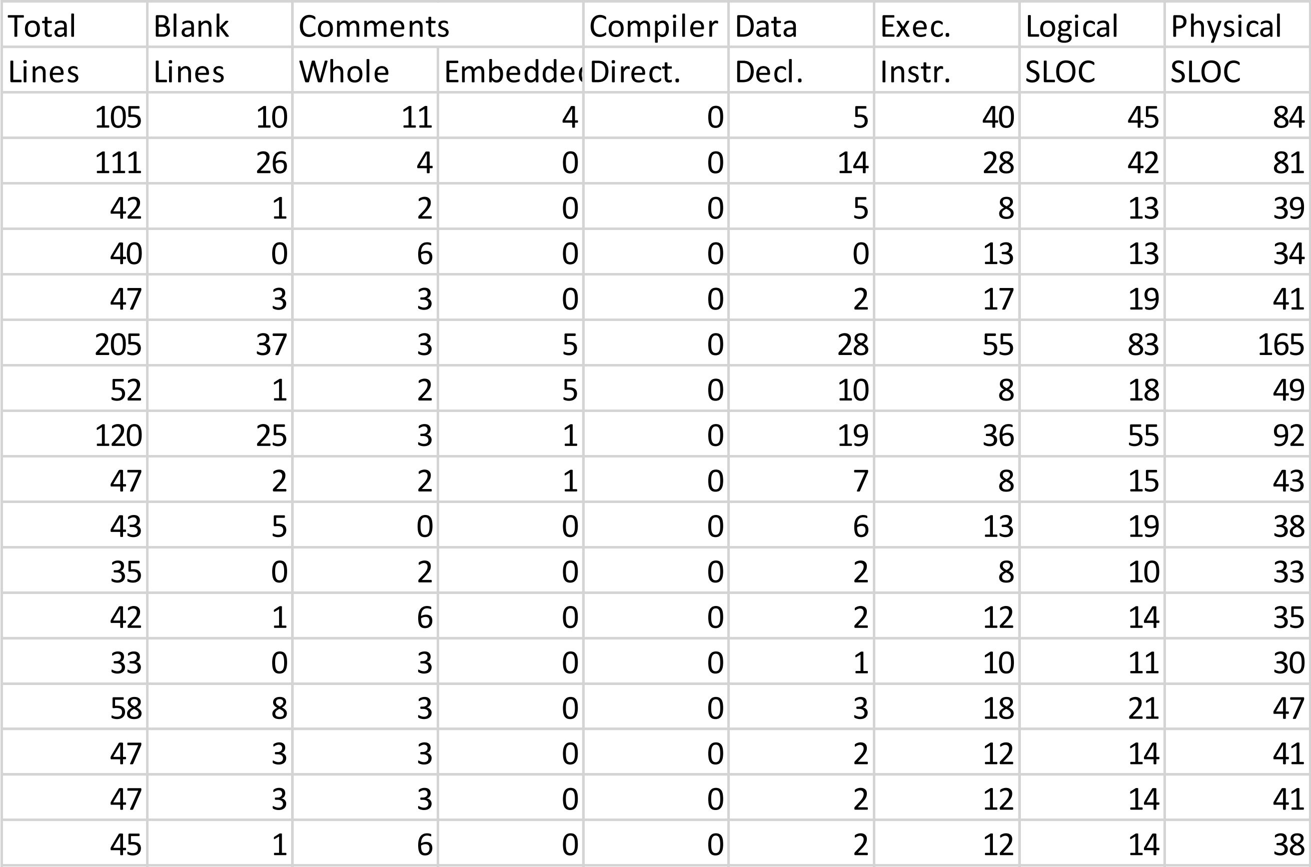
Table 4. Logical SLOC Counting Rules for C/C++, Java, and C#

Structure	Order of Precedence	Logical SLOC Rules
SELECTION STATEMENTS: *if, else if,* else, "?" operator, *try , catch, switch*	1	Count once per each occurrence. Nested statements are counted in the similar fashion.
ITERATION STATEMENTS: *For, while, do..while*	2	Count once per each occurrence. Initialization, condition and increment within the "*for*" construct are not counted. i.e. for (i= 0; i < 5; i++)… In addition, any optional expressions within the "*for*" construct are not counted either, e.g. for (i = 0, j = 5; i < 5, j > 0; i++, j--)…

CodeCount™ SLOC Tools

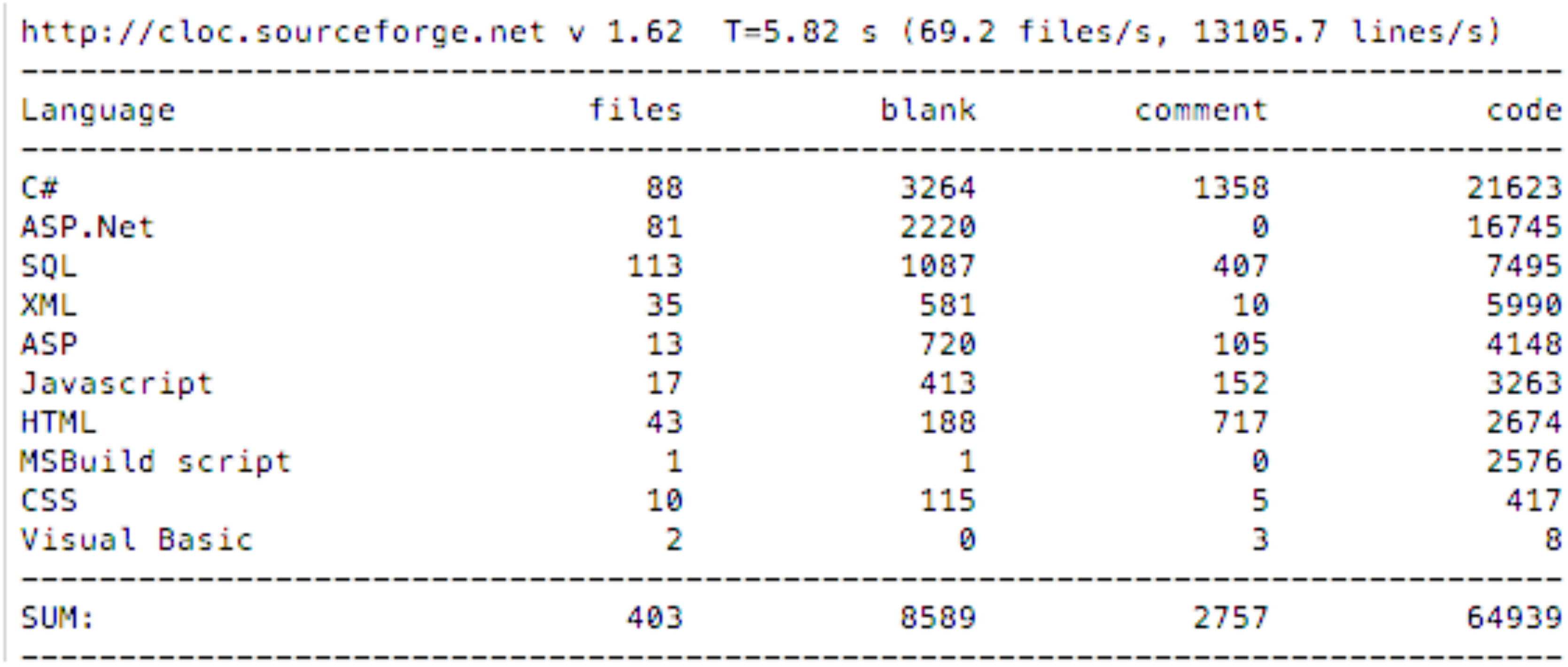
it implements the code counting framework published by the Software Engineering Institute (SEI) and adapted by COCOMO(R)

Logical vs. physical USC-UCC (http://csse.usc.edu/ucc_wp/)



Total	Blank	Comments		Compiler	Data	Exec.	Logical	Physical
Lines	Lines	Whole	Embedde	Direct.	Decl.	Instr.	SLOC	SLOC
105	10	11	4	0	5	40	45	84
111	26	4	0	0	14	28	42	81
42	1	2	0	0	5	8	13	39
40	0	6	0	0	0	13	13	34
47	3	3	0	0	2	17	19	41
205	37	3	5	0	28	55	83	165
52	1	2	5	0	10	8	18	49
120	25	3	1	0	19	36	55	92
47	2	2	1	0	7	8	15	43
43	5	0	0	0	6	13	19	38
35	0	2	0	0	2	8	10	33
42	1	6	0	0	2	12	14	35
33	0	3	0	0	1	10	11	30
58	8	3	0	0	3	18	21	47
47	3	3	0	0	2	12	14	41
47	3	3	0	0	2	12	14	41
45	1	6	0	0	2	12	14	38

Logical vs. physical : cloc



```
http://cloc.sourceforge.net v 1.62  T=5.82 s (69.2 files/s, 13105.7 lines/s)
-------------------------------------------------------------------------------
Language                     files          blank        comment           code
-------------------------------------------------------------------------------
C#                              88           3264           1358          21623
ASP.Net                         81           2220              0          16745
SQL                            113           1087            407           7495
XML                             35            581             10           5990
ASP                             13            720            105           4148
Javascript                      17            413            152           3263
HTML                            43            188            717           2674
MSBuild script                   1              1              0           2576
CSS                             10            115              5            417
Visual Basic                     2              0              3              8
-------------------------------------------------------------------------------
SUM:                           403           8589           2757          64939
-------------------------------------------------------------------------------
```

Loc Counting

- Safe when used in the same/controlled environment
- Parameters tuning
- CMMI 4 , 5