

# Functional Size measurement and COCOMO – A synergistic approach

Dr Anthony L Rollo

Software Measurement Services Ltd

## Abstract

*Accurate estimates are essential to the proper management of software development projects; many software estimates have very wide bounds between the lower and upper estimates of effort. The width of estimates bounds is determined by the degree of uncertainty in the available data. One of these sources of uncertainty is the use of previous projects; especially where those projects do not exhibit a high degree of homogeneity with the project being estimated. The most recent version of the COCOMO model enables the use of IFPUG Function Points. This is accomplished by the use of a set of Unadjusted Function Points UFP to Source Line of Code SLOC conversion tables promulgated by Software Productivity Research. Many authors consider the use of these tables may give unsatisfactory results. The reliability of the use of a single UFP/SLOC conversion figure will be explored, and shown to be subject to a great deal of variation. Additionally the conversion tables are not available for the other functional sizing methods in use today, namely MKII, NEFPUG, COSMIC-FFP.*

*The paper will propose an alternative use of the COCOMO model to assist in the task of estimation. The generally accepted method of estimation using a functional sizing method is to base the estimate on previous project data, where those projects form a homogeneous set with the project under study. The chief difficulty is to find a sufficiently homogeneous set of projects. Research previously carried out can demonstrate that by increasing the degree of homogeneity amongst a set of projects leads to a useful reduction in the variation of the estimates. The proposal is that we may sensibly use the COCOMO cost drivers to allow us to determine a set of homogeneous projects by using a technique derived from estimation by analogy. In addition the COCOMO cost drivers may be used to allow the estimator to adjust his estimates based on the differences between the cost drivers exhibited by the available data and the project under study. This paper is the result of ongoing research and it is offered as a position paper showing the results obtainable under research conditions. The author will be keen to establish links with practitioners to undertake field trials of the proposed approach.*

## 1 Introduction

Since the publication of the book [1] commonly called COCOMO II, many estimators have now used COCOMOII with a function Point (FP) input. The COCOMO model is well regarded as containing a considerable wealth of understanding of project behaviour and the factors, which affect an estimate. The publication [1] first saw the method of FP allied to this powerful model of project behaviour. However the COCOMO II model converts the FP input into Source Lines of code (SLOC) utilising a table of values associated with backfiring.

The backfiring method was proposed by Capers Jones [2] of Software Productivity Research and the method has been successfully used in sizing large portfolios of software applications. However in COCOMO II the implication is that backfiring will be used to convert the size of a single program into SLOC. Whilst this may have been implicit in the

backfiring method when it was first introduced many FP practitioners regard the method as being too inaccurate to be used on a single program or application. The backfiring method is widely used with success in the rapid sizing of portfolios of software applications.

Software Measurement Services has been successfully combining the COCOMO II model with FP in order to analyse and explain deficiencies perceived in project performance. An example of this would be that a project is perceived to have underperformed in terms of its productivity. The use of the COCOMO II model allows an exploration of such factors as schedule compression and their likely effect upon project performance. For this purpose a more accurate local conversion into lines of code may be obtained by calculating, from the total SLOC size and the measured FP size, what SLOC/FP ratio was achieved by the project. The purpose of the investigation is to highlight reasons for poor productivity and not to make accurate estimates of cost, so any variation caused by the use of a SLOC-FP conversion ratio can be ignored as the demonstration of the trend effect of a factor can still be usefully explored.

SLOC Note the definition of SLOC used within this paper is that originally proposed by Boehm in COCOMO II namely a logical source statement, excluding comments see [1] for a fuller description

### **1.1 Structure of paper**

The paper will present a few apparently disconnected subjects before drawing them together into a proposed method for integrating COCOMOII and Functional size measurement.

The paper will start with a discussion of the reliability, of FP/SLOC conversion as advocated by the backfiring method of calculating FP size, and used in the COCOMOII method.

This will be followed by a discussion of the value of a homogenous data set for estimating purposes and showing that a single previous data point is sufficient for a reliable estimate if it is sufficiently homogenous to the project being estimated.

## **2 Reliability of SLOC/FP conversion**

An exploration of the amount of variation between the numbers of SLOC generated by a project of a known FP size has been undertaken [3] and the results of this exploration are presented.

Two function point counters undertook a count of several programs written in COBOL, these programs had been classified as small medium and large (on the basis of SLOC) to assist in planning the counts. However during the counts it emerged that several programs when actually sized did not fit their original classification. As a result we had the SLOC re-counted. This also raised the question, in our minds, as to the likely accuracy of a SLOC based FP size (backfired FP). This question was later analysed in some depth [3] though the results were unpublished. I have replicated some of the original analysis for the purposes of this paper.

The population contained 20 applications and were all sized in IFPUG 4.0 on the instruction of the client. The current version of IFPUG is 4.2, however IFPUG claim that there is no statistically significant difference in size as a result of the changes from 4.0 to 4.2.

The various applications were of two type's batch and on-line and were all from the same company and the same site. The applications were concerned with stock control and monitoring, service delivery and various invoicing matters as well as a data base reconciliation program. Initially the analysis was conducted on all of the applications ignoring the difference between batch and on-line they were all written in ANSI COBOL.

Table 1 illustrates the full range of the programs and a glance at the data will reveal that there is a wide disparity between them when comparing the ratio of SLOC to FP this calculation was carried out for both the unadjusted function points (UFP) and adjusted FP (AFP) counts. Interestingly the effect of the Value adjustment factor was to reduce the size of most of the application programs.

Name	Type b/o	UFP Size	AFP Size	SLOC	SLOC/UFP	SLOC/AFP
APP1	b	10	8	1620	162.0	202.5
APP2	b	53	37	3571	67.4	96.5
APP3	b	91	61	6253	68.7	102.5
APP4	o	46	33	7552	164.2	228.8
APP5	b	79	58	11153	141.2	192.3
APP6	o	176	125	11275	64.1	90.2
APP7	b	56	39	12564	224.4	322.2
APP8	o	87	87	16703	192.0	192.0
APP9	b	468	309	18074	38.6	58.5
APP10	o	126	99	20890	165.8	211.0
APP11	b	467	308	22309	47.8	72.4
APP12	o	129	117	24070	186.6	205.7
APP13	b	335	248	36174	108.0	145.9
APP14	o	280	255	40959	146.3	160.6
APP15	o	328	257	56963	173.7	221.6
APP16	b	472	350	58804	124.6	168.0
APP17	o	354	344	61764	174.5	179.5
APP18	o	629	648	130794	207.9	201.8
APP19	o	1010	1043	143305	141.9	137.4
APP20	o	584	549	211020	361.3	384.4

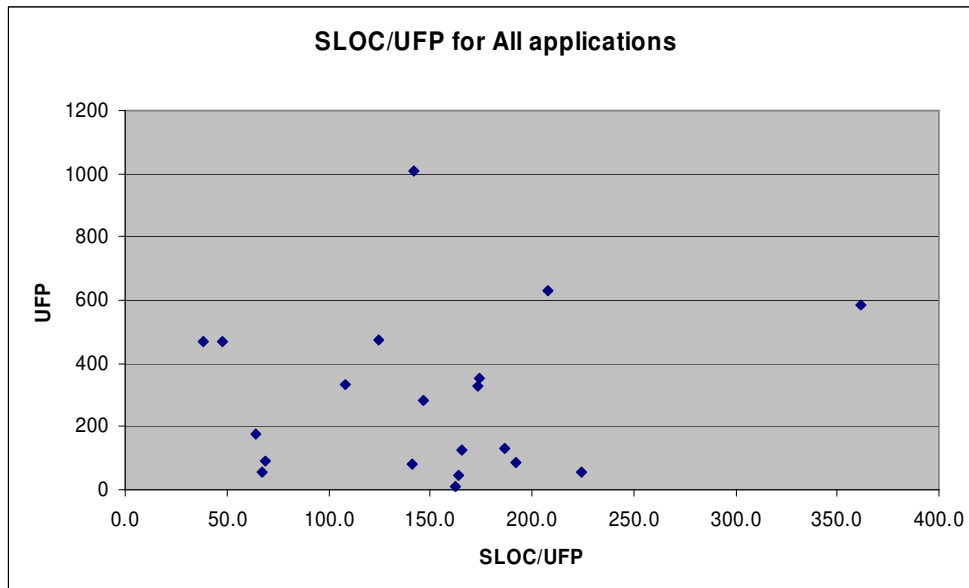
**Table 1 Function Point and SLOC sizes of 20 Application Programs**

This initial analysis was reinforced by the use of some basic statistics which are presented in table 2.

Statistic	SLOC/UFP	SLOC/AFP
Average	148.0	178.7
Stdev	74.2	79.4
Median	154.1	185.8
10th Percentile	62.4	88.4
90th Percentile	209.6	238.2
Max	361.3	384.4
Min	38.6	58.5
Ratio Max/Min	9.4	6.6

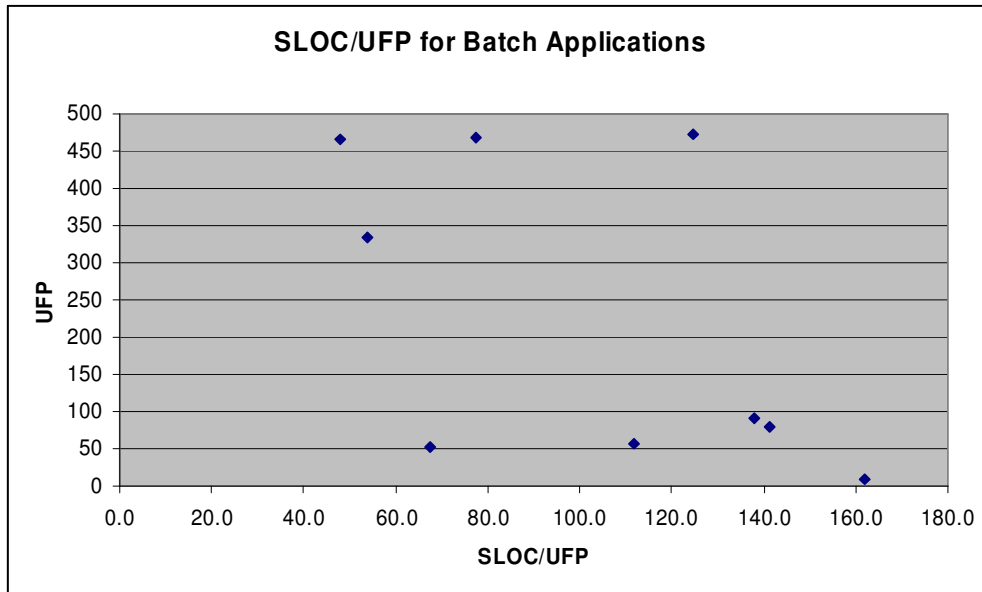
**Table 2 Statistics on the 20 Application Programs**

These statistics reveal a considerable variation in the number of SLOC generated per function point in the data we see that though the data are relatively normal in distribution the standard deviation is quite large being around half of the average indicating a considerable variation. And indeed the difference between the minimum and maximum values is a ratio of 9.4:1 and 6.6:1 respectively for UFP and AFP sizes. A Graph of UFP against SLOC is produced as

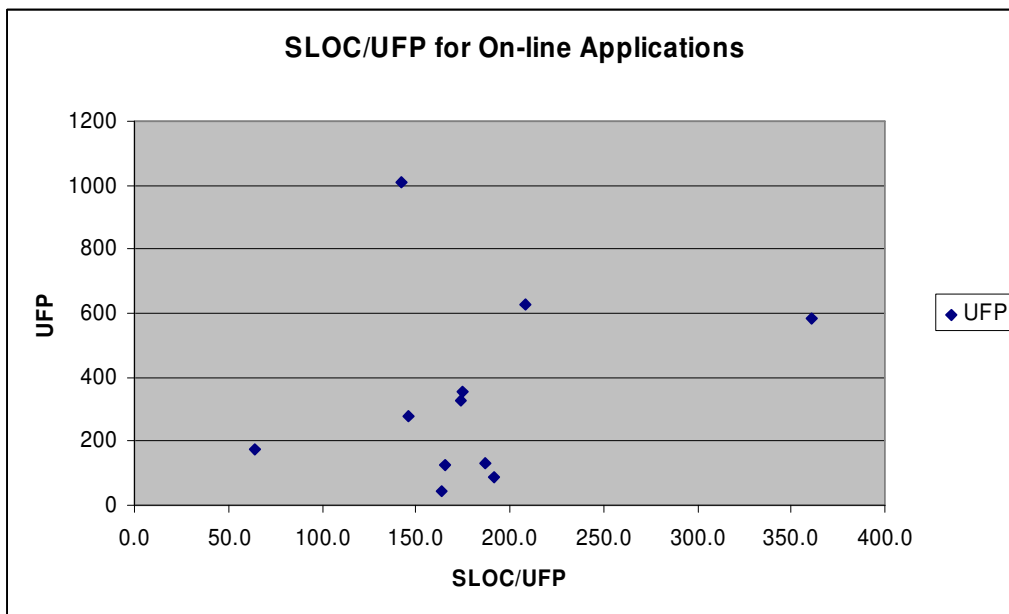


**Figure 1 graph of the number of SLOC generated per UFP for all applications**

It may be thought that fig. 1 shows us two separate groups of data with in a trend in both of them. Consequently the data were separated into their types namely batch and on-line the results are shown in fig. 2 and fig. 3 respectively.



**Figure 2 graph of the number of SLOC generated per UFP for Batch applications**



**Figure 2 graph of the number of SLOC generated per UFP for Batch applications**

As can be seen there is no real relationship between UFP size and the number of SLOC generated per UFP. Considering whether the situation would be improved if we were to consider the number of SLOC generated per Adjusted Function Point it is simply noted that the statistics show us that we have similar variation in SLOC/FP regardless of whether we use AFP or UFP. To reinforce this correlation factors were calculated and the following values obtained. As these clearly show while SLOC correlates quite well with both UFP and AFP

	UFP	AFP	SLOC	SLOC/UFP	SLOC/AFP
UFP	1				
AFP	0.996492	1			
SLOC	0.838985	0.82289	1		
SLOC/UFP	0.244887	0.235761	0.699001	1	
SLOC/AFP	0.062231	0.042585	0.550704	0.948626	1

**Table 3 Correlation coefficients**

There is no correlation between either UFP or AFP and the number of SLOC generated per Function Point. This result demonstrates that using an incorrect SLOC/FP figure as an input to the COCOMO II model can lead to considerable error in the estimate that is produced. As can be seen in table 4 the effect upon an estimate is substantial and should only be used with caution.

500 FP application		Effort in Staff Months			Likley	Likley
SLOC/FP	SLOC	optimistic	Likely	Pessimistic	Sched	Staff
39	19500	62	77	96	14.6	5.3
148	74000	267	334	417	23.3	14.3
361	180500	712	890	1113	31.8	28

*Table 4 different estimates for 500FP using different SLOC/FP ratios*

### 3 Homogeneity and estimation

The generally accepted advice is that in order to obtain an estimate the best data to use is historical data derived from the same environment as the proposed development. It is relatively easy to demonstrate that as homogeneity increases then the variation in the data tends to diminish. Taking this a stage further it is possible to assert that in many situations small data sets are to be preferred to large data sets. This is especially true where the small set is homogeneous, since a large set of data from several projects is very likely to be heterogeneous. That is, with a large data set the development environment will tend to differ across the range of projects.

An illustration of this is the data set used by Boehm (1984) which reveals a large variation in productivity. However, variation can be reduced if the data is grouped according to the factors shown in Table 1. Taking the data in Table 1 we have a fairly diverse set of data, despite the data all being from scientific applications. For example in the case of organic mode scientific applications of similar size (in thousands of Delivered Source Instructions KDSI) and written in FORTRAN under the same environmental factors (TOOL (the extent to which software tools are used, and MODP Modular Programming, being similar - between 1.1 and 0.95) the range of productivity becomes 302 - 723 actual Delivered Source Instructions (DSI) per man month, rather than 47 - 1250 DSI/MM when the full range of scientific applications is considered. The point to note is that by using a small homogeneous data set then variety is reduced and forecasts may be made with more confidence.

<b>Data Extracted From the COCOMO Data Set</b>							
<b>All Scientific Applications (Boehm, 1981. pp 496, 497)</b>							
Project number	Year	Lang.	MODP	TOOL	Mode	Size KDSI	Prod DSI/MM
31	75	MOL	1.0	1.0	E	50	47
32	72	FTN	1.10	1.10	SD	261	372
33	76	FTN	0.91	0.91	E	40	66
34	77	FTN	0.91	1.10	E	22	66
35	68	MOL	1.24	1.24	E	13	159
36	79	FTN	1.0	0.91	SD	12	218
37	78	FTN	1.0	1.0	ORG	34	723
38	77	PL1	.82	.91	ORG	15	1250
39	64	FTN	0.91	1.10	ORG	6.2	775
40	74	MOL	1.10	1.0	ORG	2.5	312
41	76	FTN	0.91	0.91	ORG	5.3	883
42	78	FTN	0.95	0.95	ORG	19.5	433
43	78	FTN	1.0	1.0	ORG	28	337
44	78	FTN	1.10	1.0	ORG	30	345
45	77	FTN	1.0	0.95	ORG	32	302
46	78	FTN	1.0	1.0	ORG	57	452
47	78	FTN	1.10	1.0	ORG	23	639

*Table 5 data extracted from the COCOMO Data Set*

Indeed where the environment is exactly the same it is possible that with only a single data point from that environment a satisfactory estimate may be produced. Let us consider some estimates that make this point.

We have two applications from a scientific data set they are both written in the same language by the same development team [5]. The requirement was to forecast the second application – position handling. However the only project available to provide us with productivity data was the oceanographic application.

<b>System</b>	<b>FP size</b>	<b>Actual Work Days</b>	<b>Actual Productivity FP/workday</b>	<b>Forecast work days</b>	<b>relative Error</b>
Oceanography	1148	3832	0.299582		
Positioning	344	1275	0.269804	1147	10%

*Table 6 estimate using one previous data point*

The previous application suggests a productivity rate of 0.3. Thus taking the size of the new application 344 we get  $344/0.3 = 1147$ . This gives an error, when compared to the actual effort used of  $(1275 - 1147)/1275 = 0.1004$  or 10%. A forecast effort that is only ten percent under the actual effort used is an acceptable error.

Another example is taken from a financial institution [5].

System	FP size	Work Days	Actual Productivity (FP/day)	Forecast work days	Adjusted estimate	Absolute relative Error
Checking	3548	3640	0.975			
Pension	1732	1319	1.313	1776	1510	15%
Share-dealing	1086	927	1.172	746 – 794	769	17%

*Table 7 a different example of the use of small data set one previous data point*

In this example we have a team who have not previously used the design methodology prior to the first project ‘Checking’. Experience suggests that we should allow for a reduction in productivity of around 35% for the first use of a new method, and around 20 % for the second project [4]. We therefore make the adjustment that, as this is the second project with the development method, we should expect an increase in productivity of around 15%, over the checking project.

The share-dealing project was undertaken by a different team who were experienced in the method so we have a non-homogenous project. However, it is a similar project in other respects, built in the same institution but by an outside team, consequently we must expect some additional effort to develop this outsourced project – experience, within our organisation is that there is a greater effort due largely to the management tasks required to coordinate the internal and external teams involved. Our experience suggests that we should allow around 10%.

The adjustment to be made is therefore to allow for a 25% increase (35%-10%) in productivity over the checking project and 10 % (20%-10%) over the pension project – this gives us an estimate of 794 work days from checking and 746 work hours from the Pension project, giving a mid point of 769. This amounts to an error of  $(927-769)/927 = 17\%$ . This would seem to be quite acceptable given the different development Team.

This has demonstrated the feasibility of obtaining an accurate estimate from a very few and even a single homogenous data point.

#### **4 Proposed use of COCOMO II with Functional Sizing**

The problem for the estimator is finding suitable historical data that can be relied upon to provide an accurate estimate of a future project. The use of COCOMO II has been seen as a way of avoiding this problem as the cost drivers allow the estimate to be adjusted to suit a variety of cost influencing factors. However as has been demonstrated the reliability of Backfiring in the context of estimating a single project is subject to a very large variation. And whilst it might be possible to adjust the language factor in COCOMO to allow for local coding practices the data, explored in Section 2 above suggests that this practice is unlikely to provide an acceptable degree of reliability.

It is proposed that a suitable approach might be to marry the normal FSM based estimate of finding a suitable project with the use of the COCOMO cost drivers to assist in determining the degree of homogeneity and as a measure of the amount of adjustment we might require in our estimate.

The COCOMO Cost drivers are compared for the Checking and Pension projects as are the share-dealing project

Cost Driver	Checking	Pension	Share-dealing
<b>Product</b>			
RELY	1.1	1.1	1.1
DATA	1.28	1.28	1.28
DOCU	1	1	1.
CPLX	1	1	1
RUSE	1	1	1.07
<b>Platform</b>			
TIME	1	1	1
STOR	1	1	1
PVOL	0.87	0.87	1
<b>Personnel</b>			
ACAP	1	1	0.85
AEXP	.88	.88	1.1
PCAP	1	1	.88
PEXP	1	1	.91
LTEX	1.2	1	.91
<b>Project</b>			
TOOL	0.9	0.9	0.9
SITE	.8	.8	1.09
SCED	1	1.14	1
EAF			
Product	1.41	1.41	1.52
Platform	0.87	0.87	1.00
Personnel	1.15	0.88	0.64
Project	0.72	0.82	0.98
Overall	1.02	0.81	0.94

*Table 8 COCOMOII Cost Driver Values*

Using these factors we now see that in order to estimate Pension we need to allow for an increase in productivity of  $1.02 - 0.81 = 0.215$  % as a percentage this is  $100 * 0.215 / 1.02 = 21\%$ :

- hence we allow a productivity of  $0.975 * 1.2 = 1.18$ ,
- this gives us  $1732 / 1.18 = 1468$  days,
- which is an error of  $1319 - 1468 = -149$  =  $149 / 1319 = 11.3\%$

System	FP size	Staff Days	Actual FP/day	Estimated FP/day	Est. work days	Absolute relative Error
Checking	3548	3640	0.975			
Pension	1732	1319	1.313	1.11	1468	11%

Share-dealing	1086	927	1.172	1.16	1031	11%
---------------	------	-----	-------	------	------	-----

*Table 9 Estimate adjustment using difference between cost multipliers*

For Share-Dealing we allow for an adjustment of  $1.02 - .94 = 0.08/1.02 = 8\%$

- this gives a productivity of  $0.975 * 1.08 = 1.053$ ,
- giving an effort of  $1086/1.053 = 1031$  staff days,
- an error of  $1031 - 927 = 11.2\%$

Both of these examples give a relative error in the estimates of 11% - it must of course be accepted that these are post-hoc estimates and therefore there is no scope creep or other similar factors to be contended with. These are clearly acceptable errors in the estimates and of course once more local data are available then degree of homogeneity and hence the accuracy of estimates can be expected to improve.

## 5 Conclusion

The paper has shown that the use of backfire FP to SLOC conversion factors is inherently inaccurate. The use of homogeneous data allows for acceptable results.

The combination of COCOMO II and functional sizing using the COCOMO II factors to calculate by how much the productivity, has been demonstrated. The estimates achieved are of a higher degree of accuracy than those achieved by using historical data and general advice and judgement to adjust the productivity figures. The estimates have an astonishingly similar degree of absolute relative error- and this seems likely to be due to chance.

It is acknowledged that this paper does not contain sufficient analysis to justify a claim that this is a more reliable method of making estimates, than currently utilised.

However it is suggestive that this might well be the case and this was a position paper

The need now is for a good deal more data to be collected and analysed before any such claim can be made.

If this method can be justified then it would apply equally well to any of the functional sizing methods as the basic estimate is based upon the historical productivity.

The author would welcome collaboration from any practitioners who have access to data which would allow this topic to be explored in greater depth.

## References

- [1] *Software Cost Estimation with COCOMOII*, Barry W Boehm et al., Prentice Hall, New Jersey, 2000
- [2] *Applied Software Measurement, Assuring Productivity and Quality*, C. Jones, McGraw Hill, New York , 1996
- [3] *Is Backfiring a useful techniques*, MSc assignment, unpublished report, H J Bush, Polytechnic University of East Anglia, 1997
- [4] *'Estimating with MKII function point analysis'* SSADM Version 4 Subject guide, CCTA publication HMSO London.
- [5] *'An Exploration of Parametric Software Cost Estimating Models for Jackson Systems Development'*, A. L. Rollo, PhD Thesis University of Aston in Birmingham, 1995