Personal Project Estimation Tool

Group ?

Simon Newton
Poya Manouchehri
Steven Kah Hien Wong

Version 1.0

About Group ?

The group name “Group ?” represents the uncertainty experienced by our group members during the initial stages of the project, as to what group we were actually in. The team started out as half of Group G, which was split when it reached a size of eight. Four members were taken a put into Group H, and by the end of week two we were down to three members. After possibly losing a second group member, plans were put in place disband the team. However, the members felt that much had already been achieved, the group continued giving rise to our unique group name.

About the Logo

The logo, drawn by Steven Kah Hien Wong, uses the question mark contained in a jail as a metaphor for containing (reducing) the uncertainty in the estimation process. To be useful, the estimation procedure must be as objective as possible to reduce the influences of bias.

Revision History

Date	Person	Change
1 Apr	Poya Manouchehri	Added feature point templates
3 Apr	Simon Newton	Added COCOMO II
8 Apr	Simon Newton	Separated COCOMO II into templates and user guide
15 Apr	Steven Kah Hien Wong	Added PERT template, PIR table and brief user guide
16 Apr	Poya Manouchehri	Modified feature point templates. Replaced IFPUG Value Adjustment Factor (VAF) with Complexity Multiplier.
17 Apr	Simon Newton	Added value for Java FP to KSLOC convertion
19 Apr	Poya Manouchehri	Added feature point user guide and PIR
19 Apr	Simon Newton	Modified COCOMO II templates and user guide
20 Apr	Simon Newton	Added risk analysis section
20 Apr	Steven Kah Hien Wong	Added full PERT user guide
21 Apr	Steven Kah Hien Wong	Modified the PERT template and PIR
22 Apr	Poya Manouchehri	Added preface section
25 Apr	Poya Manouchehri	Added feature point checklist
26 Apr	Simon Newton	Modified risk analysis section. Moved from estimating confidence interval to a separate process.
26 Apr	Poya Manouchehri	Added process combination template and PIR
27 Apr	Steven Kah Hien Wong	Added PERT checklist
29 Apr	Simon Newton	Added confidence interval to COCOMO guide and templates
29 Apr	Simon Newton	Added disclaimer
29 Apr	Simon Newton	Edited and revised section 1
1 May	Simon Newton	Modified the PMAT factor for COCOMO
8 May	Simon Newton	Added COCOMO and Combination Checklists
8 May	Simon Newton	Revised Risk Management Section
10 May	Poya Manouchehri	Revised effort and duration section of feature point templates and user guide.
10 May	Poya Manouchehri	Modified PIR section for feature points
12 May	Poya Manouchehri	Added new items to feature points checklist
14 May	Simon Newton	Added COCOMO PIR
15 May	Steven Kah Hien Wong	Added Team Size to PERT template
15 May	Steven Kah Hien Wong	Converted PERT estimation units from Months to Days
20 May	Simon Newton	Fixed some descriptions in the COCOMO guide
20 May	Simon Newton	Revised Risk Management section (again).
22 May	Simon Newton	Added example
27 May	Simon Newton	Added References Section
27 May	Simon Newton	Added information and checklists on code counting
29 May	Poya Manouchehri	Revised Feature Point PIR
29 May	Simon Newton	Added Normal Table (Appendix A)
29 May	Poya Manouchehri	Added combination PIR description
30 May	Simon Newton	Added Review Triggers
30 May	Steven Kah Hien Wong	Glossary
31 May	Poya Manouchehri	Added to references
31 May	Steven Kah Hien Wong	Finalised Page, Section and Figure Numbering
31 May	All	Proof read final document

Contents

1. Preface

1.1 The Problem

One of the main characteristics of a good engineer is the ability to manage the development of a project around a schedule and budget. In almost all cases, the clients or customers require the engineering group to produce an estimate of the duration and cost of the development, at early stages of the project, as a basis for an agreement or contract. Hence, the task of estimating project cost and duration, is one of the key responsibilities of a team manager.

While estimation is strongly tied to uncertainty, for many engineering disciplines it can be carried out accurately and reliably, for example in civil engineering. This is primarily due to the fact that the effort required for such engineering tasks tend to scale linearly with size. For instance if a one kilometre road costs $C and takes T months to finish, a ten kilometre road over similar terrain will have a cost of approximately $C × 10 and will take in the order of T × 10 months to complete. In addition, one can obtain a value representing the size of the project (in this case, the length of the road) at the beginning of the project. Furthermore, this value is unlikely to change by a large amount (the length of the road is relatively fixed, barring any obstacles that must be avoided).

In the field of software engineering however, estimation is a significantly harder task. Firstly, there is no direct, objective measure for the size of a software project. Even the most obvious metric, the number of lines of source code, isn't a precise measure as it is dependant upon the implementation language (imagine building a road where the required length depended upon what method was used to construct the road!). Therefore, metrics that do exist to tend to be proxies for the actual size, such as function points, or entity counts.

Secondly, past data indicates that the effort and cost does not scale linearly with size. Factors such as integration costs and module testing means that the effort required to build a system is often much larger than the effort required to build the separate components.

Software systems tend to undergo significantly more change during their development than any other types of systems. This is known as requirement creep or "creeping featurism". This is brought about not only from the clients (asking for more features), but also by the programmers themselves, as they want to add "that one last feature". A manager wanting to deliver on time and budget must keep both parties in check.

Finally, the end point of a software engineering project is rarely defined. Most projects end through negotiation, with the project manager and the client sitting down and discussing what has been delivered, and what remains unimplemented. In terms of estimation, we are trying to estimate an end point, which in reality doesn't exist.

The difficulty in software project estimation can be expressed as:

Not only are our goal posts moving, but they're getting further away and are invisible!

1.2 Available Tools

Despite the seemingly impossible situation faced by those whose task it is to accurately estimate software projects, many important tools and techniques have been developed. Usually these methods involve some form of measuring the “size” of the software, and then arriving at a figure for effort and duration. It is also common practice to subdivide the project into smaller, independent deliverables, which may be estimated separately. However, the warning above about the cost of the software whole being greater than the sum of the parts still holds true.

Some of the approaches to estimating software size include:

Memory Usage. Possibly the oldest metric used to determine a program’s size. The amount of memory required to execute the program was measured. For simple programs, this could often be calculated by counting the number and types of variables in scope at any time during the execution. Using memory usage as a proxy for size has fallen out of favour as projects have become significantly more complex and memory has increased from the low kilobytes to the hundreds of megabytes.

Kilo Source Lines of Code (KSLOC). Lines of source code is the intuitive way of measuring the size of software. Unfortunately like all size metrics it has limitations. As with memory usage, the actual KSLOC value is only known once the project has been delivered. Secondly, the values for lines of code is highly dependent upon the programming language used. In addition to this, KSLOC is far from an objective measure. For instance, what constitutes a line of code? Are comments counted? What about begin and end blocks (or braces in C) ? Organisations relying on lines of code as a measure of size need to have a well defined checklist of what is and isn’t included.

Function Points: Function Point Analysis and its derivative methods, are currently one of the best size metrics available to software engineers. A function point is a small unit of functionality within the software system. Function points can be calculated by counting components such as number of user interfaces that the system will have, or how many input files there will be. Since these components are part of the problem domain, rather than the solution (unlike size metrics such as KSLOC), function points can be counted with little uncertainty. Some derivatives of Function Points, are Feature Points (used in this estimation tool) and 3D Function Points. Note that function point analysis only produces an estimate of the project’s size, and still needs to be converted to effort and duration.

As with calculating size, there are a number of methods that can be used to estimate effort and duration. Some of these rely on having previously calculated the size of the system (such as COCOMO) while others rely on experience from past projects.

Algorithmic: These methods use some sort of a mathematical expression, in order to predict the duration (or cost) of a project, based on its size and other input parameters such as complexity of the system and team dynamics. Putnam’s SLIM and Boehm’s COCOMO and COCOMO II models are examples of algorithmic estimation methods.

Analogy: This involves deriving the duration directly from previous projects, using multiplicative and additive factors. Given *n* previous projects, the estimate may look something like

$$Estimate = \frac{1}{n}\sum_{i=1}^{n}(Mf_i \bullet Actual_i + Af_i)$$

where Mf_i and Af_i are the multiplicative and additive factors, respectively. Estimation from analogy requires the existence of reliable data from past projects, preferable from within the same organisation.

Expert Opinion: One of the most widely used estimation methods is expert opinion. This typically involves breaking the project into smaller parts and having experts make estimations on each. These estimates are used to calculate a mean and variance (confidence interval). PERT and Delphi methods are examples of expert opinion estimation. This has the advantage of not requiring information from previous projects, but instead relying on the experts experience to guide them in the estimation process.

1.3 Process Overview

This estimation tool can be separated into three separate processes:

1) The actual estimation of duration using three methods, Feature Point Analysis, a refined COCOMO II method, and PERT (a form of expert opinion). An overview of this process is shown in Figure 1.1 on the next page and a description of each method follows.

   Feature Point Analysis
   A feature point analysis is carried out which gives a measure for the size of the software. It is then converted into effort using past data (or supplied default values) for hours per feature point. A confidence interval is then calculated, and the values for pessimistic effort, expected effort and optimistic effort are converted into estimates for duration.

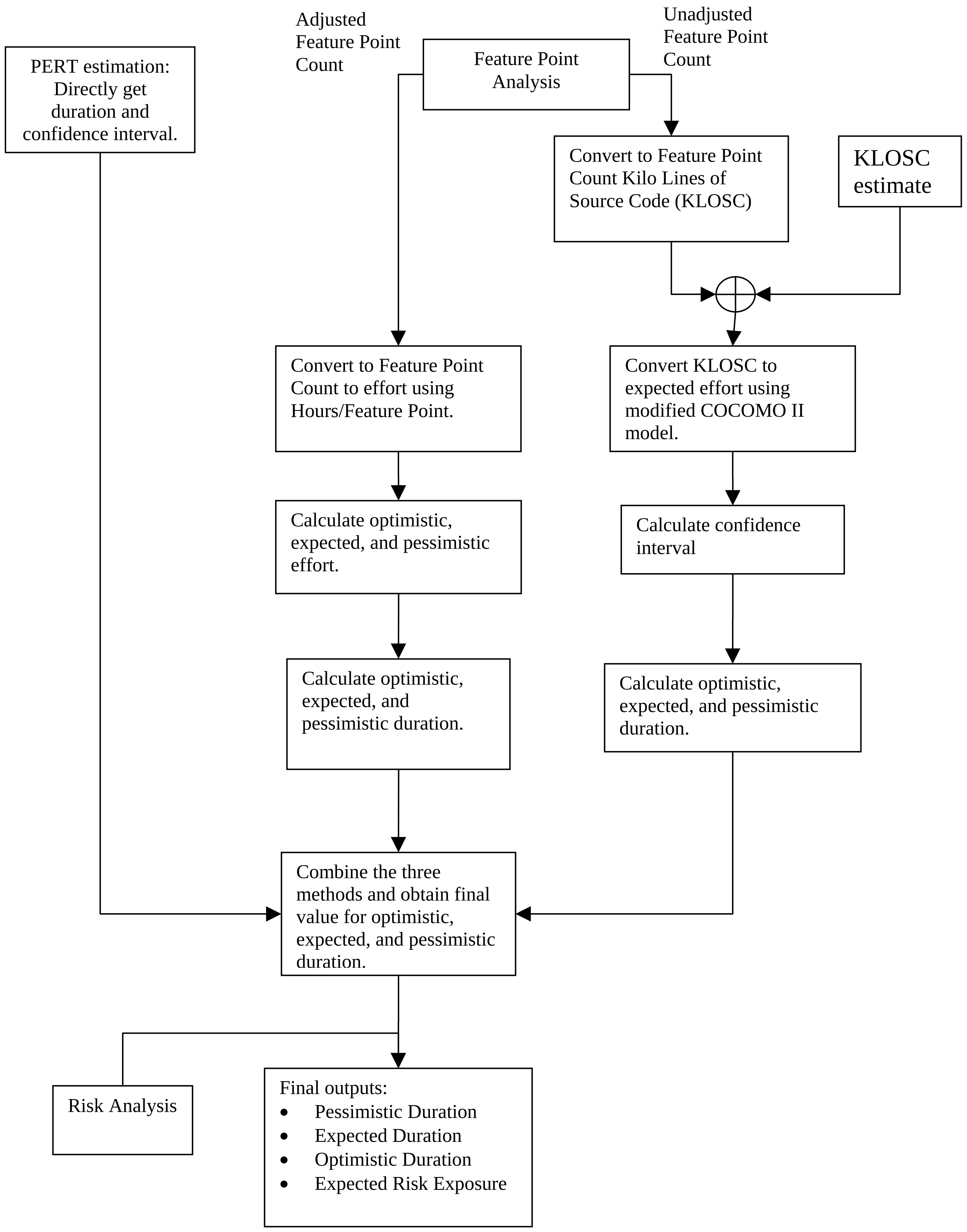
   COCOMO II
   The unadjusted feature point count can be used as the input for the COCOMO II model or alternatively an estimate of size in Kilo-Lines-of-Source-Code KSLOC can be made using Delphi techniques. Once the size is determined, the COCOMO II process is used to obtain an estimate for the amount of effort that will be required for the project. A confidence interval is then calculated, depending on the level of confidence required, and the stage the project is at. This is then converted into an estimate for the pessimistic, expected and optimistic duration.

   PERT
   PERT is quite independent from the other two methods. It directly gives a estimation of duration, as well as variance and confidence interval using expert opinion.

   The final step in this process involves combining the three duration estimates, taking into account the performance of each method in previous projects. A risk analysis of the project is also carried out which is used to give an estimate for the amount of additional effort required in the project.

2) The second process involves a set of review triggers that see the estimation process described above being repeated, as new information about the problem or solution become available. Reviews must be carried out at the suggested stages, in order to ensure that the estimated duration and the confidence interval always reflect all the available information.

The Estimation Process



PERT estimation: Directly get duration and confidence interval.
Adjusted Feature Point Count
Feature Point Analysis
Unadjusted Feature Point Count
Convert to Feature Point Count Kilo Lines of Source Code (KLOSC)
KLOSC estimate
Convert to Feature Point Count to effort using Hours/Feature Point.
Convert KLOSC to expected effort using modified COCOMO II model.
Calculate optimistic, expected, and pessimistic effort.
Calculate confidence interval
Calculate optimistic, expected, and pessimistic duration.
Calculate optimistic, expected, and pessimistic duration.
Combine the three methods and obtain final value for optimistic, expected, and pessimistic duration.
Risk Analysis
Final outputs:
Pessimistic Duration
Expected Duration
Optimistic Duration
Expected Risk Exposure

Figure 1.1. An overview of the estimation process

3) The third and final process is the post implementation review (PIR). This is a critical part of the estimation tool, as it uses information obtained from past projects to refine future estimations. There are four distinct PIRs that are carried out in this process. The first three refine each of the estimation methods in the first section (ie Feature Point Analysis, COCOMO II, and PERT). The final PIR refines the weights that are used to combine the estimations produced by these three methods. That is, the method(s) that had performed better in estimating durations for past projects, will be given a higher weighting for future estimations.

1.4 Assumptions and Constraints

This estimation tool is subject to the following set of constraints and assumptions:

- The estimation tool uses a feature point analysis as part of the estimation process. It is therefore assumed to be suitable for both transactional systems and real-time applications that are algorithm intensive.

- We have assumed that Feature Points are close enough to Function Points, as to be used for size measurements under the COCOMO II model.

- That the people using this tool have a reasonable knowledge of the estimation process, and have past experience either managing or working on software projects.

1.5 Guidelines on the Effective Use of the PPET

The following are a set of general guidelines that will aid in the use of this tool and help to improve the accuracy of the estimations.

- Each estimation will need a copy of the templates. It is recommended that a photocopy be made of the following sections
    - Section 3
    - Section 5

- Many templates have spaces provided for comments. Use these to provide justification for the values obtained. This is particularly important when subjective decisions must be made such as choosing the rating for the COCOMO factors. Descriptive comments will assist both you and others in the future to learn from the estimates. Remember, what is obvious now, will be forgotten tomorrow – *write it down*.

- Divide the project into as many *independent* sub-tasks as possible. This is specifically useful for PERT estimation, as the experts are more likely to produce better estimates for smaller tasks.

- Use ALL three methods of estimations available in this tool and combine them as described. Relying on a single estimation method is risky at best.

- After carrying out an estimation, make sure that a different person signs off the work by going through the checklists available. This ensures that there are no simple errors in calculations, and that the original estimator has not missed any critical information.

- Review your estimations whenever a factor that may affect the estimate changes. The section on Review Triggers (Section 4), describes in more detail when the estimations should be reviewed. This ensures that your duration estimations always reflect the latest information about the state of the project. It also gives you a more definitive confidence interval as more knowledge about the problem and/or solution is gained.

- As mentioned before, Post Implementation Reviews (PIRs) are critical to improving the estimations. Processes such as COCOMO rely heavily on calibration to a specific organisation. It is expected that the estimates produced by the initial versions of this tool will vary substantially from actual values until enough data has been collected to calibrate the tool. In a similar manner, attempting to calibrate the tool using data from projects of vastly different size and domains will not produce reliable estimates.

- Once both the estimation and checklists have been completed, they should be archived for future reference. They will be needed again for the PIR, and should be kept as they provide valuable past data and insight into the estimation process.

2.0 User Guides

2.1 Feature Point Analysis User Guide

2.1.1 How to use the Templates

In order to make an estimate of effort and duration using feature point count, the feature point templates in section 3.1 need to be completed. The following steps describe how the templates should be filled:

1) The upper portion of the main template (Figure 3.1) consists of six sections, namely Algorithms, External Inputs, External Outputs, External Inquiries, Internal Logic Files, and External Interface files. Each of these component types is described in more detail in section 2.1.2 . You must identify the number of each of these components in your project.

2) After identifying the components, you must categorise them, based on their complexity. Section 2.1.3 is a guideline for assigning complexities to the components.

3) Now fill the upper section of the main template by writing the number of each of the components, under the appropriate complexity column. Multiply the numbers with their specified weights and write the result in the space provided. For each component type, sum the counts for low, average, and high complexity columns, and write the result in the right hand column (Total).

4) Sum the totals to get the *unadjusted feature points*. Write this value, labeled [A1], in the appropriate box.

5) In the complexity multiplier template (Section 3.1.2), pick an appropriate number for *Problem Complexity* and *Data Complexity* (from 1 to 5). Add these two numbers to get a value between 2 and 10. Finally use this sum to look up an appropriate weight in Figure 3.2 (the weight is between 0.6 and 1.4). This is the complexity factor. Write it in the appropriate box (labeled [A2]) in the Figure 3.1.

6) In the main template, multiply the unadjusted feature points with the complexity factor, to obtain the adjusted feature point count (labeled [A3]).

7) The next step is converting the adjusted feature points to values for optimistic, expected, and pessimistic effort, in section 3.1.3. To do this, first write down the number of effective working hours per month (HPM) for you organization. The default value of HPM is 85 (5 hours per day, 17 days per month). Also obtain the value of MMRE and R(C) from the PIR section 6.1 . Note that

   - R(C) is the equation for *hours per feature point*, calculated in the PIR for feature points (Section 6.1). If no past project data are available, use the model described in section 2.1.4
   - MMRE is the *mean magnitude of relative error*, also calculated in the PIR for feature points (Section 6.1). If no past project data are available, set the MMRE to 0.
   - For optimistic and pessimistic effort estimates, we deviate from the actual feature point count by ±12%, which reflects uncertainty in the accuracy of the count and is taken from Kemerer (1993)

- The $\frac{1}{HPM}$ factor converts the effort from person hours, to person months

Now we can calculate each of optimistic, expected, and pessimistic effort values by using the following equations:

$$\text{Effort}_{\text{optimistic}} = [A3] \times 0.88 \times (1 - MMRE) \times R([A3] \times 0.88) \times \frac{1}{HPM}$$

$$\text{Effort}_{\text{expected}} = [A3] \times R([A3]) \times \frac{1}{HPM}$$

$$\text{Effort}_{\text{pessimistic}} = [A3] \times 1.12 \times (1 + MMRE) \times R([A3] \times 1.12) \times \frac{1}{HPM}$$

8) The final step is converting the values obtained for effort, to durations (section 3.1.4). We simply use an exponential model, taken from original COCOMO, to do this conversion. Please refer to section 2.1.5 for details of converting effort to duration.

2.1.2 Component Definitions

***Algorithm* –** is the set of rules, which must be completely expressed to solve a significant computational problem. For example, a square root extraction routine, or a Julian date conversion routine, would both be considered algorithms.

External Input (EI) - is an elementary process in which data crosses the boundary from outside to inside. This data may come from a data input screen, electronically or another application. The data can be either control information or business information. If the data is business information it is used to maintain one or more internal logical files. If the data is control information it does not have to update an internal logical file.

External Output (EO) - an elementary process in which derived data passes across the boundary from inside to outside. The data creates reports or output files sent to other applications. These reports and files are created from one or more internal logical files and external interface file .

Derived Data is data that is processed beyond direct retrieval and editing of information from internal logical files or external interface files. Derived data is the result of algorithms, and/or calculations. *Derived data occurs when one or more data elements are combined with a formula to generate or derive an additional data element(s).*

External Inquiry (EQ) - is an elementary process with both input and output components that result in data retrieval from one or more internal logical files and external interface files. This information is sent outside the application boundary. The input process does not update any Internal Logical Files and the output side does not contain derived data.

***Internal Logical File (ILF)* –** is a user identifiable group of logically related data that resides entirely within the applications boundary and is maintained through External Inputs.

External Interface File (EIF) - a user identifiable group of logically related data that is used for reference purposes only. The data resides entirely outside the application and is maintained by another application. The External Interface File is an Internal Logical File for another application.

2.1.3 Component Complexity Guidelines:

Algorithm:

Algorithm Type	**Complexity**
Simple [eg. A = B + C*(D-E)] or moderate expressions [eg. D = SQRT(B*2-4*A*C)]	Low
Standard math and statistical routings. Basic matrix/vector operations. Basic numerical analysis. Ordinary differential equations. Basic truncation and round off.	Average
Difficult numerical analysis. Analysis of noisy, stochastic data. Partial differential equations. Parallelization.	High

Figure 2.1.

External Input:

	Data Element Types		
File Types Referenced	**1 - 4**	**5 - 15**	**> 15**
0 - 1	Low	Low	Average
2	Low	Average	High
> 2	Average	High	High

Figure 2.2.

External Output, External Inquiry:

	Data Element Types		
File Types Referenced	**1 - 5**	**6 - 19**	**> 19**
0 - 1	Low	Low	Average
2 - 3	Low	Average	High
> 3	Average	High	High

Figure 2.3.

Internal Logic Files, External Interface Files:

	Data Element Types		
Record Element Types	**1 - 19**	**20 - 50**	**> 50**
1	Low	Low	Average
2 - 5	Low	Average	High
> 5	Average	High	High

Figure 2.4.

2.1.4 Hours Per Feature Point

If no data from previous projects are available, use the following equation to calculate the value of R (hours per feature point). This equation is based on data collected from nearly 1,000 projects from over 100 companies, available on the IFPUG website (Longstreet, 2003). It is however strongly recommended that the equation for R be based on your company's past projects, as it is a highly organizational specific factor.

$$R(C) = 1.41e^{0.0003049 \bullet C}$$

If the above formula is used, the MMRE should be set to 0. This does NOT indicate the lack of uncertainty in the estimate, rather that no confidence interval can be reliably predicted due to the lack of past data.

2.1.5 Duration

To convert the effort obtained in section 3.1.3 to duration, use the following equation:

Duration in Months = $\mathbf{a} \times \text{Effort}^{\mathbf{b}}$

Where the values of **a** and **b** are selected from the following table:

Development Mode	**a**	**b**
Organic	2.5	0.38
Semidetached	2.5	0.35
Embedded	2.5	0.32

Figure 2.5.

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. Example: An accounting system.

Semidetached Mode – The project's characteristics are intermediate between Organic and Embedded

Embedded Mode – The project is chacterised by tight, inflexible constraints and interface requirements. An embedded mode project will require a great deal of innovation. Example: A real-time system with timing constraints and customised hardware.

2.2 COCOMO II User Guide

This section deals with producing an effort and duration estimate using a modified COCOMO II model.

- Estimate the size using lines of source code
- Calculate the scale factor B
- Calculate the cost driver multiplier M
- Calculate the expected effort and duration

2.2.1 Estimating Size

The COCOMO II model requires an estimate for the size of the system in thousands of lines of source code (KSLOC). This can be produced by either directly estimating the size of modules, or by converting the feature point count to KSLOC by using a lookup table. Early in the development process, when system design has not been completed, it is recommended to use the unadjusted feature point count to produce an estimate for the size of the system. Once design has been completed, and the number and scope of modules defined, the size (in KSLOC) of each module can be estimated directly.

It is strongly recommended that the first time this tool is used, the organisation develops a standard for counting lines of source code. A checklist has been provided in Appendix B, along with details on how it is to be used. Appendix C provides a sample of what a completed checklist might look like. For more information on counting lines of code see *Software Size Measurement: A Framework for Counting Source Statements* (Park 1992).

2.2.1.1 Using Unadjusted Feature Points

Copy the figure for unadjusted feature points obtained in Figure 3.1 into the field marked [B1]. Multiply [B1], by the figure obtained from Figure 2.6 for the specified development language, and enter the result into [B2].

Language	**SLOC / UFP**
4GL (average)	15
Ada	71
AI Shell	49
APL	32
Assembly	320
Assembly (Macro)	213
ANSI / Quick / Turbo Basic	64
Basic - Compiled	91
Basic - Interpreted	128
C	128
C++	29
ANSI Cobol 85	91
Fortran 77	105
Forth	64
Jovial	105
Lips	64
Java	23
Modula 2	80
Pascal	91
Prolog	64
Report Generator	80
Smalltalk	20
Spreadsheet	8
SQL	12
Visual Basic	32

Figure 2.6. Used to convert between feature points and lines of source code (Boehm 2000)

Divide [B2] by 1000, to get an estimate for the size of the system in thousands of lines of source code, and enter the result into [B3].

Proceed to section 2.2.2 to calculate the scale factor

2.2.1.2 Direct Estimation using Simplified Delphi Techniques

As previously mentioned, this approach requires the number and scope of modules to be defined, and hence requires system level design to be complete. Note here that the word module is used in a implementation independent manner. It could refer to classes in Java, modules in Perl, or files in C.

Each module should be listed in the first column of Figure 3.3 (add more rows if required). For each module, multiple experts should be approached and, on being supplied with the module specification, asked to estimate the number of kilo lines of code it would take to implement the module. These experts would typically be the developers responsible for the implementation of the module, as they would have first hand knowledge of the relative size of the module.

It is very important that these experts be kept isolated and do not communicate their estimate to one another. The estimates are collected, and entered into Figure 3.4. Once all estimates are in, the average is calculated and entered into the final column of Figure 3.4.

The average for each module is then returned to the experts, and they are given the opportunity to revise their estimate. The new estimates are then entered into Figure 3.4, and again the average is calculated. Finally the sum of all the averages is obtained, and entered into the cell marked [B3].

2.2.2 Calculation of Scaling Factor (β)

This reflects the dis-economies of scale associated with a software project. That is if the size of the system doubles, the amount of effort required increases by a factor larger than 2. This represents in increased costs in managing team members, communication overheads etc.

Five factors affect the scaling driver. A description of each is given below, followed by a table which lists the choices and the weights. For the factors, select the choice that most closely matches the situation faced by the project. Then enter the appropriate weight into Figure 3.5 in the template section.

Precedentedness (PREC)

This value models the organisation's past experience with projects of a similar type. The range is from very low (5) indicating the organisation has no previous experience with a project of this type, to extremely high, which would indicate that the organisation has an in depth understanding of the application domain.

Questions to be considered include :

- How well are the product objectives understood?
- What is the organisation's experience with related software?
- Will this project require the concurrent development of new hardware and/or operating procedures?
- Will the project require new and innovative data processing algorithms?

	Very Low	**Low**	**Nominal**	**High**	**Very High**	**Extremely High**
Precedentedness (PREC)	thoroughly unprecedented	largely unprecedented	somewhat unprecedented	generally familiar	largely familiar	throughly familiar
Value	**5**	**4**	**3**	**2**	**1**	**0**

Figure 2.7

Development Flexibility (FLEX)

This models the flexibility of the development process. If the client specifies only the general goals of the project (ie little client intervention in the development process) a value of extremely high should be used. However, if the development process is completely specified and rigorously controlled by the client, a value of very low should be used.

Questions that should be considered include:

- Is there a need for software conformance with pre-established requirements?
- Is there a need for software conformance with external interface specifications?
- Is there a premium on early completion of the project?

	Very Low	**Low**	**Nominal**	**High**	**Very High**	**Extremely High**
Development Flexibility (FLEX)	rigorous	occasional relaxation	some relaxation	general conformity	some conformity	general goals
Values	**5**	**4**	**3**	**2**	**1**	**0**

Figure 2.8

<u>Risk Resolution (RESL)</u>

This reflects the amount of risk analysis performed by the organisation. Consider the following questions:

- Has a risk identification and analysis been performed? Section 2.4 should be used as a basis for this.
- Have milestones been set to revolve important risk items?
- Are the risks being managed? Have contingency plans been developed for the most important risks?
- Is a risk monitoring procedure in place? If so, is it performed regularly?
- Are there tools available for monitoring and resolving risk items?

	Very Low	**Low**	**Nominal**	**High**	**Very High**	**Extremely High**
Risk Resolution (RESL)	little (20%)	some (40%)	often (60%)	generally (75%)	mostly (90%)	full (100%)
Values	**5**	**4**	**3**	**2**	**1**	**0**

Figure 2.9

<u>Team Cohesion (TEAM)</u>

This accounts for how well the development team work together. A value of extremely high reflects a integrated team that understand each other well. A value of very low would be used for a team that has difficult or minimal interactions.

Factors to consider include:

- Consistency of members objectives and cultures
- Ability and willingness of members to accommodate other members objectives
- Experience of members in operating as a team
- Extent of team building exercises undertaken

	Very Low	**Low**	**Nominal**	**High**	**Very High**	**Extremely High**
Team Cohesion (TEAM)	very difficult interactions	some difficult interactions	basically cooperative interactions	largely cooperative	highly cooperative	seamless interactions
Values	**5**	**4**	**3**	**2**	**1**	**0**

Figure 2.10

Process Maturity Factor (PMAT)

This rates the maturity of the software development process within the organisation. It is based on the Software Engineering Institute's Capability Maturity Model (CMM) which defined five levels of maturity for the software development process. It is expected that the organisation has a good understanding as to which level of the CMM it is operating at. The levels are briefly described below but for a more detailed description of the CMM, see Paulk 1993.

Initial Level. At this level, an organisation does not have effective management procedures or project plans. If formal procedures for project control exist, there are no organisational mechanisms to ensure that they are used consistently. The organisation may successfully develop software but the characteristics of the software (quality etc.) and the process (budget, schedule etc.) will be unpredictable.

Repeatable Level. At this level, the organisation has formal management, quality assurance and configuration control procedures in place. It is called the repeatable level because the organisation can successfully repeat projects of the same type. However there is a lack of a formal process model. Project success is dependent on individual managers motivating a team and on organizational folklore acting as an intuitive process description.

Defined level. At this level, an organisation has a defined its process and thus has a basis for qualitative process improvement. Formal procedures are in place to ensure that the defined process is followed in all software engineering projects.

Managed level. A level 4 organisation has a defined process and a formal program of quantitative data collection. Process and product metrics are collected and fed into the process improvement activity.

Optimising level. At this level, an organisation is committed to continuous process improvement. Process improvement is budgeted and planned and is an integral part of the organisation's process

Process Maturity Factor (PMAT)	Initial	Repeatable	Defined	Managed	Optimizing	
Value	**4**	**3**	**2**	**1**	**0**	

Figure 2.11

Calculation of Scaling Factor B

Sum the PREC, FLEX, RESL, TEAM and PMAT factors, and enter the result into [B4]

Calculate the value of the scaling factor β, by using the following formula, and enter the result into [B5]

$$\beta = 1.01 + 0.01 \times [B4]$$

2.2.3 Cost Drivers

COCOMO II uses sixteen cost drivers to refine the estimate. Each of these are described below, and a weight should be chosen for each, and entered into Figure 3.6. In the case where a driver does not seem appropriate, the weight of 1.0 should be used, although note that the accuracy of the estimate will be comprised by omitting factors.

Required Software Reliability (RELY)

This factor models the required reliability of the software. It is determined by estimating the impact that a software failure would cause.

RELY	slight inconvenience	low, easily recoverable	moderate, easily recoverable	high financial loss	risk to human life
Weight	0.75	0.88	1.0	1.15	1.39

Figure 2.12

Database size (DATA)

This factor considers the effect that large data sets will have on development time by calculating the a value for D/P. Large data sets typically cause an increased in development time due to the large number of test cases that must be produced.

The D/P factor is measured in bytes per line of code. It is calculated using the following:

D/P = Database Size (Bytes) / Program Size

Where program size is the value obtained for B3 in section 3.2.1 (from either feature points or lines of code). Note that the above formula deals with **lines of code** and not kilolines of code.

The value for D/P is then used to determine the value for the DATA factor from the table:

DATA	$D/P < 10$	$10 < D/P < 100$	$100 < D/P < 1000$	$D/P > 1000$
Weight	0.93	1.0	1.09	1.19

Figure 2.13

Documentation match to life-cycle needs (DOCU)

This factor accounts for the extra effort involved in producing documentation. It is estimated by comparing the level of documentation needed for the development process to that which will be produced. For example, documentation that leaves many of the life cycle uncovered (sparse documentation) would be assigned a ranking of very low. On the other hand excessive documentation, that more than covers the needs of the project life cycle would be rated as very high.

DOCU	Many needs left uncovered	Some needs left uncovered	Right sized for project life cycle	Excessive for project life cycle needs	Very excessive for project life cycle needs
Weight	0.89	0.95	1.0	1.06	1.13

Figure 2.14

Required Re-usability (RUSE)

Increased effort will be required to develop reusable components due to standardizing interfaces, increased documentation and more elaborate testing. The RUSE factor represents this increased effort, by rating the level of re-usability required.

RUSE	none	across project	across program	across product line	across multiple product lines
Weight	0.91	1.0	1.14	1.29	1.49

Figure 2.15

Execution Time Constraint (TIME)

This driver accounts for the increased effort involved in meeting execution time constraints when developing software. It is determined by estimating the percentage of system execution time that will be used by the software under development.

TIME	< 50% of available time	70%	85%	95%
Weight	1.0	1.11	1.31	1.67

Figure 2.16

Main Storage Constraint (STOR)

The STOR represents the percentage of total system storage resources consumed by the software system.

STOR	< 50%	70%	85%	95%
Weight	1.0	1.06	1.21	1.57

Figure 2.17

Platform Volatility (PVOL)

This factor accounts for possible change in the underlying platform that the software is being developed on. The platform could include: hardware, operating system, device drivers, libraries, compilers etc.

PVOL	major: 12 mon minor: 1 mon	major: 6 mon minor: 2 wk	major: 2 mon minor: 1 wk	major: 2 wk minor: 2 days
Weight	0.87	1.0	1.15	1.30

Figure 2.18

Analyst Capability (ACAP)

The factor accounts for the capability of the analysts designing the software. This us done by estimating the percentile that the analyst team resides in. This ranking should consider factors such as analysis and design ability, efficiency and thoroughness and the ability to communicate and cooperate. This should not take into account experience which is rated using AEXP. An extremely strong analyst team would be ranked in the 90th percentile.

ACAP	15th percentile	35th percentile	55th percentile	75th percentile	90th percentile
Weight	1.5	1.22	1.0	0.83	0.67

Figure 2.19

Programmer Capability (PCAP)

This reflects the capability level of the programmers. Programmers should be considered as a team and not as individuals. Factors that should be considered include: efficiency and thoroughness and ability to communicate and cooperate. Once again programmer experience should not be considered here, as PEXP accounts for this.

PCAP	15th percentile	35th percentile	55th percentile	75th percentile	90th percentile
Weight	1.37	1.16	1.0	0.87	0.74

Figure 2.20

Personal Continuity (PCON)

This accounts for the turnover rate of the organisation in terms of the percentage of staff per year.

PCAP	48% / year	24% / year	12% / year	6% / year	3% / year
Weight	1.26	1.1	1.0	0.91	0.83

Figure 2.21

Applications Experience (AEXP)

This represents the level of experience the development team has with this type of application. It ranges from very low (less than 2 months) to very high (over 6 years).

AEXP	< 2 months	6 months	1 year	3 years	6 years
Weight	1.23	1.1	1.0	0.88	0.80

Figure 2.22

Platform Experience (PEXP)

This factor accounts for the level of experience that the development team has with the particular platform. The platform should be the same set of hardware/software that was considered in the PVOL factor. A team with less than 2 months experience would be rated very low, while a team with over 6 years would receive a rating of very high.

PEXP	< 2 months	6 months	1 year	3 years	6 years
Weight	1.26	1.12	1.0	0.88	0.80

Figure 2.23

Language and Tool Experience (LTEX)

This factor reflects the experience of the development team with the programming language and software engineering tools. This includes but is not limited to : design and analysis tools, compilers, libraries and configuration management software.

LTEX	< 2 months	6 months	1 year	3 years	6 years
Weight	1.26	1.11	1.0	0.91	0.83

Figure 2.24

Software Tools (TOOL)

This factor accounts for the capability of the software development tools. The low end of the spectrum is a simple code editor, with no integration with the system design tools. The upper end on the other hand is where the software tools are closely linked with the software development process

TOOL	edit, code, debug	simple front end, back end CASE, little integration	basic life cycle tools, moderate integration	strong, mature life cycle tools, moderate integration	strong, mature pro-active life cycle tools, well integrated with development process
Weight	1.2	1.1	1.0	0.88	0.75

Figure 2.25

Multi-site Development (SITE)

This reflects the extra effort involved with multi site developments, and the effect of restrictied communication channels. It is found by calculating the average of the collocation and communication factors.

The weight for the SITE factor should be taken as the average of the collocation and communication factors.

Collocation	International	Multi-city and multi - company	Multi-city or multi company	Same city or metro area	Same building	Fully collocation
Communications	Some phone, mail	Individual phone, Fax	Narrow-band email	Wide-band electronic comms.	Wide-band elect comms. occasional video conf.	Interactive multimedia
Weights	1.24	1.1	1.0	0.92	0.85	0.79

Figure 2.26

Product Complexity Driver (CPLX)

The product complexity driver measures the inherent complexity built into the product. This is divided up into five areas: control operations, computation operations, device dependant operations, data management operations and user interface.

The CPLX factor is the average of the five areas.

	Very Low	**Low**	**Nominal**	**High**	**Very High**	**Extremely High**
Control Operations	Straight code with few nested blocks. Simple module composition	Straightforward nesting	Mostly simple nesting. Some intermodule control. Simple callbacks or message passing.	Highly nested programming operators. Queue and stack control. Distributed programming	Reentrant and recursive coding. Task synchronization and complex callbacks.	Multiple resource scheduling with dynamically changing priorities.
Computational Operations	Simple expressions eg. A = B + C*(D-E)	Moderate expressions eg. D=SQRT(B**2-4*A*C)	Standard math and statistical routings. Basic matrix / vector operations	Basic numerical analysis. Ordinary differential equations. Basic truncation, roundoff concerns	Difficult but structured numerical analysis, partial differential equations. Simple parallelization	Difficult and unstructured analysis. Analysis of noisy, stochastic data. Complex parallelization
Device-dependent Operations	Simple read, write statements with simple formats	No knowledge of particular device needed. IO done at GET/PUT level	I/O includes device selection, status and error checking	Operations at physical IO level (seeks, reads, etc). Optimised IO overlap	Routines for interrupt diagnostics, servicing and masking. Performance intensive embedded systems	Device timing dependant coding. Performance critical embedded systems
Data Management Operations	Simple arrays in main memory. Simple database queries and updates	Single file sub-setting with no data structure changes. Moderately complex queries and updates.	Multi-file input and single file output. Simple structural changes and edits. Complex DB queries and updates	Simple triggers archived by data stream contents. Complex data restructuring	Distributed database coordination. Complex triggers. Search optimisation.	Highly coupled dynamic relational and object object structures.
UI Management Operations	Simple input forms and report generators	Use of simple GUI builders	Simple use of widget set	Widget set development and extension. Simple voice I/O, multimedia	Moderately complex 2D/3D, dynamic graphics and/or multimedia	Complex multimedia and/or virtual reality
Weights	0.75	0.88	1.0	1.15	1.30	1.49

Figure 2.27

Calculation of Product Multiplier (M)

Calculate the product of the sixteen cost drivers, and enter the result into the field [B6].

2.2.4 Expected Effort

Calculate the estimate of the amount of effort required for the product, using the following formula:

$$\text{Effort} = A \times [B3]^{[B5]} \times [B6]$$

where [B3] is size in kilo lines of code, from either section 3.2.1. The value of A should be determined from past projects. In the case where no past data is available, a value of 3.0 should be used.

2.2.5 Confidence Interval

The COCOMO II model includes a sample of projects, where the size estimates are compared to the final size values, at various stages during project development.

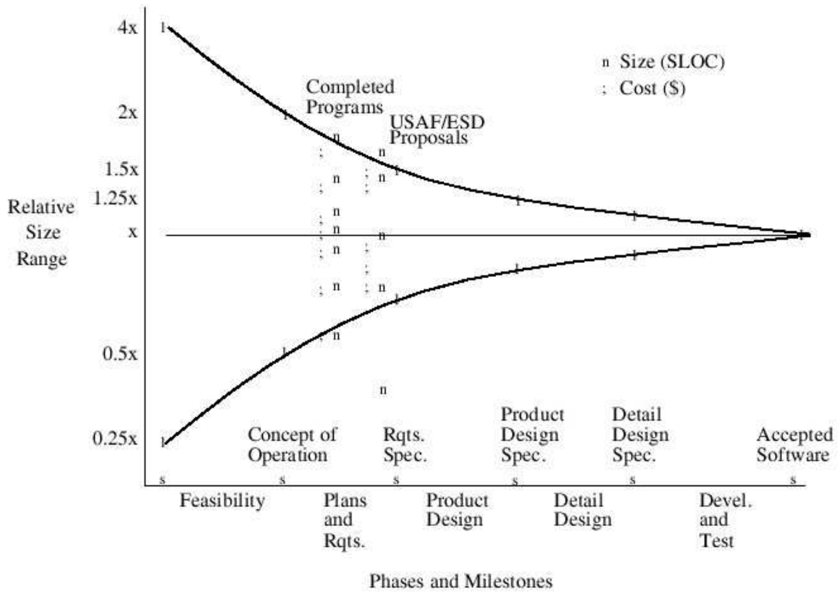


Figure 2.28

From the above graph and the COCOMO II paper by Boehm (1995), the following optimistic and pessimistic values are obtained. This represents one standard deviation about the mean (E) which, in this case, is the estimate of the effort provided by the COCOMO II algorithm.

Stage	**Optimistic**	**Pessimistic**
Concept Document Completed	0.5 E	2.0 E
Requirements Specification Complete	0.67 E	1.5 E
Architecture Design Complete	0.80 E	1.25 E
Detailed Design Complete	0.9 E	1.1 E

Figure 2.29

Note that the distribution of projects is non-symmetrical. Although there is an equal chance of a project finishing before or after the mean, the overruns and under-runs will not always balance out. The deviation from the mean for projects that finish early will in general be a lot smaller than the deviation for projects that finish late. This is reflected in the values in Figure 2.9.

An interval of one standard deviation either side of the mean gives an approximately 68% confidence interval. In order to determine a 95% confidence interval (or an interval to an arbitrary level of confidence for that matter), the procedure below should be followed.

Choose a value γ, which is the level of confidence required (ie. a value of 0.95 means we are 95% certain that the actual time taken will reside within our confidence interval). Note that a confidence of 100% cannot be obtained, and so setting γ to 1 is not possible, We then calculate the area under the tail of the normal curve using the following formula:

$$\alpha = (1 - \gamma) / 2$$

Using the example above with $\gamma = 0.95$, $\alpha = 0.025$.

Next, the values of z_L and z_H must be calculated. This gives the cutoff values for the selected confidence interval for a standardised normal distribution. These values are calculated from the table of cumulative probabilities for the normal distribution given in Appendix A. z_L is found by locating the value of α in the table, and noting for what value of z it occurs at. ZH is found by locating the value $1-\alpha$ in the table, and again noting for what value of z this occurs at. z_L should always be less than 0, while z_H should be greater than 0.

For the example of $\alpha = 0.025$, we find $z_L = -1.96$ and $z_H = 1.96$.

Finally we calculate the optimistic and pessimistic estimates for the effort using the following formula:

$$E_O = E (1 + \sigma_L . z_L)$$

$$E_P = E (1+ \sigma_H . z_H)$$

Where σ_L and σ_H are chosen from Figure 2.30, depending on what stage the project is at.

Stage	$\boldsymbol{\sigma_L}$	$\boldsymbol{\sigma_H}$
Concept Document Completed	0.5	1.0
Requirements Specification Complete	0.33	0.5
Architecture Design Complete	0.2	0.25
Detailed Design Complete	0.1	0.1

Figure 2.30

Note that for high confidence factors (such as more than 95%) and early stages, the value for E_O can become less than 0. Obviously this is not possible and the large confidence interval represents the high degree of uncertainty inherent in projects during the early stages. For this reason, it is recommended that the confidence factor (γ) be kept to less than 70% until the requirement specifications are complete.

2.2.6 Duration

Convert the expected effort, pessimistic effort E_P and optimistic effort E_O to duration using the following formula:

$$\text{Duration} = A \times E^{0.33 + 0.2 \times ([B5] - 1.01)}$$

Again the value of A is organisation specific. If no past data exists, use a value of 3.0.

These three values are then used in section 2.5, along with the results from the other two processes, to produce a final estimate.

2.3 PERT Estimation User Guide

1. Divide the project into Estimable Work Units (EWU). These are like modules. They should be small and self-contained, with clear objectives and deliverables.
2. For each EWU, one or more *Individual Estimation Form*s must be filled out. Each estimator can only fill out one form. The more estimators, the better - anyone with at least basic estimation knowledge can contribute. A description for each field on the form is as follows:
   - <u>Work Unit Name</u>

     The name of the work unit this estimation is for.
   - <u>Estimator Name</u>

     The estimator's name, so each estimation can be traced back to their original estimator.
   - <u>Team Size</u>
     Number of people that will be working on this work unit.
   - <u>Realistic Estimate</u> (R)

     An estimate for the duration of this work unit, in days, if it progresses normally.
   - <u>Optimistic Estimate</u> (O)

     An estimate for the duration of this work, in days, if it progresses well.
   - <u>Pessimistic Estimate</u> (P)

     An estimate for the duration of this work, in days, if it progresses poorly.
3. After collecting all *Individual Estimator Forms*, record all estimates in the *Work Unit Estimate Forms*. There must be one of these forms for each EWU. A description of each field in the form is below:
   - <u>Work Unit Name</u>

     The name of the Estimable Work Unit this form is for.
   - <u>Estimator Name</u>

     Name of the estimator, who's estimates will be filled out on the same row.
   - <u>Optimistic</u> (O)

     Estimator's optimistic estimate value.
   - <u>Realistic</u> (R)

     Estimator's realistic estimate value.
   - <u>Pessimistic</u> (P)

     Estimator's pessimistic estimate value.
   - <u>Unadjusted, Weighted Estimate</u> (E_u)

     The weighted average of the optimistic, realistic, and pessimistic estimates. It is calculated with the following formula:

     $$E_u = \frac{O + 4R + P}{6}$$
   - <u>Unadjusted Variance</u> (σ_u^2)

     The variance between the optimistic and pessimistic estimates. It is calculated with:

     $$\sigma_u^2 = \left(\frac{P - O}{6}\right)^2$$

- Estimator Confidence (K)

  A modifier used on the final estimate. This value is based on the estimator's previous estimates. See the *Estimator History Form* for details on how its value is found. If there is no existing estimation history, then just use the value 1.
  A value of one is for accurate estimators (their estimates will be unmodified). Values higher than one are for optimistic estimators (tend to be below actual durations). Lower than one is for pessimistic estimators (tend to be higher than actual durations). New estimators start with a confidence value of 1, which means their estimates will be unmodified.
  The confidence can be calculated from previous estimation history, ideally recorded on the *Estimator History Form*.
- Adjusted, Weighted Estimate (E_a)

  The Estimator Confidence is applied onto the Weighted, Unadjusted Estimate to find this value:
  $$E_a = K \times E_u$$
- Adjusted Variance (σ_a^2)

  The Estimator Confidence is applied onto the Unadjusted Variance to find this value. The confidence needs to be squared, because the variance is also a squared value:
  $$\sigma_a^2 = \sigma_u^2 \times K^2$$
- Average, Adjusted Estimate (E_v)

  The average of all adjusted estimates calculated:
  $$E_v = \frac{\sum E_a}{n}$$
- Average, Adjusted Variance (σ_v^2)

  The average of all adjusted variances calculated:
  $$\sigma_v^2 = \frac{\sum \sigma_a^2}{n}$$

4. After all Average, Adjusted Estimates, and Average, Adjusted Variances have been collected, record them down in the *Project Estimate Form*. This form consists of the following fields:

   - Work Unit Name

     The name of the Estimable Work Unit, which will have its estimate and variance recorded on the same row.
   - Average, Adjusted Estimate (E_v)

     The value of Average, Adjusted Estimate for the work unit specified on the same row.
   - Average, Adjusted Variance (σ_v^2)

     The value of Average, Adjusted Variance for the work unit specified on the same row.
   - Total Project Variance (E)

     The sum of all Average, Adjusted Estimates. It is the final estimate of the project:

$$E = \sum E_v$$

- <u>Total Project Variance</u> (σ^2)

  The sum of all Average, Adjusted Variances. It is the final variance of the project:

$$\sigma^2 = \sum \sigma_v^2$$

- <u>Estimate Standard Deviation</u>
  The standard deviation of the estimates:

$$\sigma = \sqrt{\sigma^2}$$

5. Finally calculate the values of optimistic, expected and pessimistic duration for the project. We assume that the duration probability of the project has a normal distribution, and take 2 standard deviation on either side of the expected value to give us a 95% confidence interval:

   Duration $_{\text{optimistic}} = E - 2\sigma$
   Duration $_{\text{expected}} = E$
   Duration $_{\text{pessimistic}} = E + 2\sigma$

   Convert the durations from Days to Months, by dividing them by the average number of working days per month for your organisation. Typically this is about 17 days per month.

2.4 Risk Management

This section deals with identifying risks associated with the project, and then analysing them to produce a measurement for the amount of risk associated with the project.

- Identify the risks
- Assign probabilities and effort values to the risks
- Calculate Risk Exposure (RE) for each risk
- Compute the Mean Risk Exposure (MRE)

2.4.1 Identify the Risks

This can be performed using a brainstorming session with stakeholders from the project. Risks from each of the five sections (technology, people, organisational, tools and requirements) should be considered. Once the risks are identified, they should be entered into Figure 3.7 in section 3.4. Note that only risks that are being borne by the organisation are entered. If risks have been written out of contracts, or passed on to sub contractors, they should not be examined in this case. In other words, it is expected that a risk analysis has been performed, and the critical risks have already been mitigated.

The five areas to consider when identifying risks are from Sommerville (2001):

Technology Risks - Risks that are derived from the software or hardware technologies that are used as components of the system being developed.

eg. A database that cannot perform as many transactions as required.

People Risks - Risks that are associated with the people developing the product.

eg. The lead programmer leaving the team.

Organisational - Risks which derive from the organisational environment in which the software is being developed.

eg. The organisation is restructured so that a new management team is responsible for the project.

Tools - Risks that derive from the CASE tools and other support software used to develop the system.

eg. The CASE tools have limitations that were not known at the beginning of the project.

Requirements - Risks that derive from changes to the customer requirements and the process of managing requirements change.

eg. The client changes the requirements to include new features.

Note that the sixth area proposed by Sommerville, *estimation*, has been omitted. For this tool, the level of uncertainty in the estimates is reflected in the confidence interval, rather than the risk exposure.

2.4.2 Assign Values

Each risk identified in step one should now be assigned a probability of occurring (between 0 and 100%). Figure 2.30 should be used as a guide, though the probabilities do not have to be the exactly values as given in the table.

Very Low	**Low**	**Moderate**	**High**	**Very High**	**Extremely High**
5%	20%	40%	65%	75%	90%

Figure 2.30. Guide to calculating the probability of a risk occurring.

Next, for each risk, the required additional effort that would be needed if the risk occurred is estimated (ie. the impact the risk would have on the required effort). This is in the form of person-days. The values for effort should be entered into Figure 3.7.

For example, a risk may be that a library that the software will use has not been used in the organisation before. We estimate that there is a 5% chance that the library contains serious errors, and if this occurs an additional 3 person days will be required to resolve these issues.

When assigning probabilities to risks, it is important to note that some risks may not be independent. This creates a problem in risk management as the risk exposure values can only be added if the individual risks are independent. This is a known problem with risk analysis, and as yet no simple methods have been devised to combat it. It is recommended that the estimator only consider independent events when using this risk management technique.

2.4.3 Risk Exposure

Next the Risk Exposure (RE) for each risk should be calculated using the formula given below:

Risk Exposure = Probability x Effort

This should also be entered into figure 3.7.

2.4.4 Expected Risk Exposure (ERE)

The expected risk exposure is calculated by summing each individual risk exposure and the final value is entered into the field marked sum. This represents the expected (average) overrun for the project and is used in section 2.5.

2.5 Process Combination User Guide

Once an estimation of optimistic, expected and pessimistic duration have been obtained from each of the three method (ie Feature Point Analysis, COCOMO II, and PERT), these are combined using Section 3.5.

The weighs for each method are initially 1.0, and are adjusted by the PIR.

Compare the values in the second column of Figure 3.8, which are the estimated excepted duration for the project. If any of these largely disagrees with the other two, go back and review the process of obtaining that estimation. Start by using the checklists in section 5 of the estimation tool, and do a thorough review of that particular estimation processes. If you still do not obtain a value close to the other estimation methods, consider reviewing the other methods of estimation. If this is also not helpful, leave the outlying value out of the following equations when combining the durations.

The final values for the optimistic, expected and pessimistic duration are then calculated using the following formula:

$$\text{Final Optimistic Duration} = \frac{(O1\times AW1)+(O2\times AW2)+(O3\times AW3)}{AW1+AW2+AW3}$$

$$\text{Final Expected Duration} = \frac{(E1\times AW1)+(E2\times AW2)+(E3\times AW3)}{AW1+AW2+AW3}$$

$$\text{Final Pessimistic Duration} = \frac{(P1\times AW1)+(P2\times AW2)+(P3\times AW3)}{AW1+AW2+AW3}$$

Finally copy the Expected Risk Exposure value calculated in section 3.4 into the field marked ERE. The ERE represents the expected overrun in required effort for the project. Note that this measurement does not indicate a duration

3. Templates

3.1 Feature Point Analysis Templates

3.1.1 Feature Point Count

Figure 3.1

Component	**Complexity**			
	Low	**Average**	**High**	**Total**
Algorithms	____ × 2 = ____	____ × 3 = ____	____ × 5 = ____	
External Inputs	____ × 3 = ____	____ × 4 = ____	____ × 6 = ____	
External Outputs	____ × 4 = ____	____ × 5 = ____	____ × 7 = ____	
External Inquiries	____ × 3 = ____	____ × 4 = ____	____ × 6 = ____	
Internal Logic Files	____ × 5 = ____	____ × 7 = ____	____ × 10 = ____	
External Interface Files	____ × 5 = ____	____ × 7 = ____	____ × 10 = ____	
			Total Unadjusted Feature Points [A1]	
			Complexity Multiplier [A2]	
			Total Adjusted Feature Points [A3]	

Estimator's Comments:

3.1.2 Complexity Multiplier

Problem Complexity: _____

1) Simple algorithms and simple calculations
2) Majority of simple algorithms and calculations
3) Algorithms and calculations of average complexity
4) Some difficult or complex algorithms
5) Many difficult algorithms and complex calculations

Data Complexity: _____

1) Simple data with few variables
2) Numerous variables, but simple data relationships
3) Multiple files, fields, and data intersections
4) Complex file structures and data intersections
5) Very complex file structures and data intersections

Sum of Problem and Data Complexity	2	3	4	5	6	7	8	9	10
Complexity Multiplier	0.6	0.7	0.8	0.9	1.0	1.1	1.2	1.3	1.4

Figure 3.2

Complexity Multiplier [A2] = _____

Estimator's Comments:

3.1.3 Effort

Hours Per Month (HPM) = ___________ (Default is 85)

MMRE = ___________

R(C) = ______________________________________ (Equation)

Effort $_{\text{optimistic}}$ $= [A3] \times 0.88 \times (1 - MMRE) \times R([A3] \times 0.88) \times \frac{1}{HPM}$

= ___________ Person Months

Effort $_{\text{expected}}$ $= [A3] \times R([A3]) \times \frac{1}{HPM}$

= ___________ Person Months

Effort $_{\text{pessimistic}}$ $= [A3] \times 1.12 \times (1 + MMRE) \times R([A3] \times 1.12) \times \frac{1}{HPM}$

= ___________ Person Months

Note: Refer to the PIR section 6.1 for the function of R, and value of MMRE

3.1.4 Duration

Duration = **a** Total Effort $^{\mathbf{b}}$

Duration $_{\text{optimistic}}$ = **a** (Effort $_{\text{optimistic}}$)$^{\mathbf{b}}$ = _______ Months

Duration $_{\text{expected}}$ = **a** (Effort $_{\text{expected}}$)$^{\mathbf{b}}$ = _______ Months

Duration $_{\text{pessimistic}}$ = **a** (Effort $_{\text{pessimistic}}$)$^{\mathbf{b}}$ = _______ Months

Note: See section 2.1.5 for selecting appropriate values of **a** and **b**

Estimator’s Comments:

3.2 COCOMO II - Templates

3.2.1 Estimating Size

⚠ To calculate system size, either use unadjusted function points (section 2.2.1.1) , **OR** estimate the lines of code on a per module basis (section 2.2.1.2)

Using Unadjusted Feature Points

Unadjusted function point count from Function Point Analysis = _____________ [B1]

Equivalent SLOC = __________________ [B2]

Size of System in KSLOC : ([B2] / 1000) = _________________________ [B3]

Estimator’s Comments:

Direct Estimation

All values should be entered in kilo lines of source code (KSLOC)

Module	Person A	Person B	Person C	Person D	Person E	Average

Figure 3.3 Initial estimates for module size

Module	Person A	Person B	Person C	Person D	Person E	Average
					Total [B3]	

Figure 3.4 Revised estimates for module size

3.2.2 Scale Factors

Factor	**Value**	
PREC		
FLEX		
RESL		
TEAM		
PMAT		
Sum		**[B4]**

Figure 3.5

$\beta = 1.01 + 0.01 \times [B4]$

Value of exponent β = _________________ [B5]

Estimator’s Comments:

3.2.3 Cost Drivers

Driver	**Value**
RELY	
DATA	
DOCU	
RUSE	
TIME	
STOR	
PVOL	
ACAP	
PCAP	
PCON	
AEXP	
PEXP	
LTEX	
TOOL	
SITE	
CPLX	
Product [B6]	

Figure 3.6

Estimator's Comments:

3.2.4 Expected Effort

Value of A = _____________ (from PIR or default to 3.0)

$$\text{Effort} = E = A \times [B3]^{[B5]} \times [B6]$$

Effort (Person Months) Estimate = ___________ [B7]

3.2.5 Confidence Interval

Required Confidence Level γ = _________________

Area under tail of normal curve: $\alpha = (1-\gamma)/2$ = _________________

Lower cutoff z_L = ________________

Upper cutoff z_H = ________________

Optimistic Effort E_O = __________ [B8]

Pessimistic Effort E_P = __________ [B9]

3.2.6 Duration

$$\text{Duration} = A \times E^{0.33 + 0.2 \times ([B5] - 1.01)}$$

Where E is either [B8], [B7] or [B9] for optimistic, expected and pessimistic duration respectively.

Optimistic Duration Estimate: ____________________ (months)

Expected Duration Estimate: ____________________ (months)

Pessimistic Duration Estimate: ____________________ (months)

Estimator’s Comments:

3.3 PERT Estimation Templates

3.3.1 Individual Estimator Form

Work Unit Name	
Estimator Name	

Team Size	

Estimates (Duration in Days)				
Optimistic (O)		Realistic (R)		Pessimistic (P)

3.3.2 Work Unit Estimate Form

Work Unit Name	

Estimator Name	Optimistic (O)	Realistic (R)	Pessimistic (P)	Unadjusted, Weighted Estimate (E_u)	Variance (σ_u^2)	Estimator Confidence (K)	Adjusted Estimate (E_a)	Adjusted Variance (σ_a^2)

Average, Adjusted Estimate (E_v)	
Average, Adjusted Variance (σ_v^2)	

3.3.3 Project Estimate Form

Work Unit Name	Average, Adjusted Estimate (E_v)	Average, Adjusted Variance (σ_v^2)

Total Project Estimate (E)	
Total Project Variance (σ^2)	
Estimate Standard Deviation (σ)	

Duration $_{\text{optimistic}} = E - 2\sigma$ = ____________ Days = ____________ Months

Duration $_{\text{expected}} = E$ = ____________ Days = ____________ Months

Duration $_{\text{pessimistic}} = E + 2\sigma$ = ____________ Days = ____________ Months

3.4 Risk Management Templates

Risk	Probability	Effort	RE = Probability x Effort
		ERE	

Figure 3.7 Risk Management Template

3.5 Process Combination Templates

Revision Number	
Performed By	
on	

Method	**Optimistic Estimation of Duration**	**Expected Duration**	**Pessimistic Estimation of Duration**	**Accuracy Weight**
Feature Point Analysis	[O1] =	[E1] =	[P1] =	[AW1] =
COCOMO II Analysis	[O2] =	[E2] =	[P2] =	[AW2] =
PERT Analysis	[O3] =	[E3] =	[P3] =	[AW3] =

Figure 3.8

O = **Final Optimistic Duration** = $\frac{(O1 \times AW1) + (O2 \times AW2) + (O3 \times AW3)}{AW1 + AW2 + AW3}$ = _______

E = **Final Expected Duration** = $\frac{(E1 \times AW1) + (E2 \times AW2) + (E3 \times AW3)}{AW1 + AW2 + AW3}$ = _______

P = **Final Pessimistic Duration** = $\frac{(P1 \times AW1) + (P2 \times AW2) + (P3 \times AW3)}{AW1 + AW2 + AW3}$ = _______

ERE = **Expected Risk Exposure** = ____________________________ (person –days)

Next revision due in ______________ months

Estimator’s Comments:

4.0 Estimation Review Triggers

As mentioned in section 1.5, the estimation process should be performed regularly throughout a project's life, so that the estimates are always based on up to date information. This will lead to a set of revisions in the estimation process, where each revision should produce an estimate with a higher degree of accuracy than the last.

Performing a revision is identical to performing the initial estimation. Copies should be made of the templates (section 3) and the checklists (section 5) and then completed as per the user guide. When the estimation process is complete, a different individual should user the checklists to identify any problems with the estimation.

The following events should be used to trigger an estimation review.

Problem Changes
This is the most common cause to trigger a new revision of the estimation. Whenever information about the problem changes (ie. through a change to the requirements specification), a new revision should be prepared. This will ensure that all stakeholders are aware of the effect that the changes have had on the estimate, and can often dispel the "just a few small changes" myth.

Resources Change
Any change in the resources available to a project should trigger a new revision. For example this could occur when:

- Personnel join or leave
- The development platform changes
- Specific resources become unavailable (ie testing hardware etc.)

Milestones
The completion of a project milestone should trigger a new revision. Generally at the completion of a milestone, new information about the project will become available (due to decisions having to be made in order to meet the milestone). This will lead to an increased accuracy of the next revision.

Elapsed Time
The "catch all" trigger. If no revision has been performed within a specified period of time, this should trigger a review. The period of time to perform the next revision should be set once a revision has been completed. The amount of time is dependant upon the organisation and project, though for any reasonable sized project (more than 6 months duration), the authors suggest performed at least 4 revisions over the lifetime of the project, or at least every two to three months.

5.0 Checklists

These checklists should be completed at the end of each estimation by a person other than those who performed in initial estimation. They assist in identifying errors that may have been made during the estimation process. Once complete, they should be signed, and stored with the completed estimation templates for future reference.

5.1 Feature Point Analysis Checklist

Item	**Check**
The total number of each component type (i.e. algorithms, external inputs, external outputs, external inquiries, internal logic files, and external interfaces files) has been determined.	
The components are classified as being of low, medium or high complexity according to guidelines in section 2.1.3 . Appropriate values are put in figure 3.1 .	
The raw feature points for the components are summed up correctly on the right hand side of Figure 3.1, to give Total Unadjusted Feature Point Count [A1].	
In section 3.1.2, an appropriate value between 1 and 5 is chosen for Problem Complexity and Data Complexity, and written in the provided space.	
The sum of the values for Problem and Data Complexities are used to pick the correct factor from Complexity Table in Figure 3.2. This factor is written back into Figure 3.1.1 [A2]	
The unadjusted feature point count [A1] is correctly multiplied with the complexity multiplier [A2], to produce the Total Adjusted Feature Point Count [A3].	
In section 3.1.3 a value for Hours Per Month (HPM) for the organisation is picked	
In section 3.13 the value of MMRE, and the function R(C) are correctly stated, based on the PIR for feature points (section 6.1)	
In section 3.13 values of optimistic, expected, and pessimistic duration are correctly calculated using the provided formulae and parameters.	
Durations are correctly derived from effort values in section 3.1.4	

Checked by	Signature
Date	

5.2 COCOMO II Checklist

Item	**Check**
Ensure that the size has been correctly estimated by using unadjusted feature points (section 2.2.1.1). Ensure they have been converted to KSLOC correctly, pay particular attention to the division by a factor of 1000 (ie. KSLOC rather than LOSC)	
OR	
If the size has been estimated directly, ensure that the process of estimation has been carried out as specified in section 2.2.1.2. Both Figure 3.3 and 3.4 should have been completed, using Delphi techniques. Ensure that the unit calculated for B3 are in thousands of lines of source code.	
The five scaling factors (PREC, FLEX, RESL, TEAM and PMAT) have all been calculated. Decisions have been accounted for.	
The exponent β has been calculated correctly	
Values have been assigned to the 17 cost drivers. Reasons have been given for any factors that were deemed not relevant. All decisions have been justified	
The expected effort has been calculated correctly	
The confidence factor γ has been set, a value for α calculated, the values for z_L and z_H obtained and finally values for E_P and E_O calculated.	
The expected, pessimistic and optimistic values for duration have been calculated.	

Checked by	Signature
Date	

5.3 PERT Estimation Checklist

Individual Estimator Forms

Item	**Check**
Estimable Work Units do not share any common goals. Each unit should be its own self-contained piece of work.	
Estimable Work Units are as small as feasibly possible. Smaller units are easier to estimate for.	
Each estimator’s name is filled out, so all individual estimations can be accounted for.	
No estimators have the same names. If there are any, then they should be differentiated, somehow; either by including their middle names, or appending a number to each (i.e. “John Smith #1”).	
The realistic (R), optimistic (O), and pessimistic (P) estimates are correct, relative to each other. Pessimistic should be the highest, followed by realistic, and optimistic should be the lowest. That is, the following condition should hold true: $P > R > O$	

Work Unit Estimate Forms

Item	**Check**
All the individual estimators have made their estimates for the work unit. Otherwise estimations will need to be re-processed from this step onwards, if there are any more additions or modifications to the estimates.	
The Work Unit name has been filled out and is correct. All the individual estimates put into this form must have matching Work Unit names, too.	
The estimator’s name and their corresponding estimates have been transferred properly from their individual form to the Work Unit Estimate form. The values should match.	
The Unadjusted, Weighted Estimates, and Variances have been calculated correctly.	
The value for the Estimator Confidence is the latest one used. Make sure all estimation history is recorded and up-to-date, and that the Estimator Confidence value used, is based on those values.	
The Adjusted Estimates, and Adjusted Variances have been calculated correctly.	
The Average, Adjusted Estimates and Variances have been added up and calculated correctly.	

Project Estimate Form

Item	Check
All Work Unit Estimate forms have been filled out for all Work Units	
The values from each individual Work Unit Form have been copied into the Project Estimate form correctly	
The Total Project Estimate and Total Project Variance have been added up correctly.	

Estimator History Form (PIR)

Item	Check
The Estimator Name is filled out correctly with a valid name.	
There are no duplicate forms for the same estimator. Strictly one form per estimator.	
Work Unit Names have been copied over correctly.	
Values for each Unadjusted, Weighted Estimate (E_u), and Actual Duration (A), corresponds correctly with the Work Unit Name, given on the same row.	
Difference values have been calculated correctly	
All the latest estimates and actual durations have been recorded.	
The Estimator Confidence (K) is based on all the latest values given in the form, and has also been calculated correctly.	

Checked by	Signature
Date	

5.4 Risk Management Checklist

Item	**Check**
Ensure that all five areas have been considered when identifying risks	
Check that reasonable values have been assigned to the risk probabilities	
Check that reasonable values have been assigned to the risk impacts (effort)	
Check that all risk exposures have been correctly calculated.	
Check that the expected risk exposure has been correctly calculated.	

Checked by	Signature
Date	

5.5 Process Combination Checklist

Item	**Check**
All relevant values have been copied correctly into Figure 3.8	
Any outliers have been identified, and have triggered a review of the appropriate method.	
The final values for the optimistic, expected and pessimistic duration have been calculated correctly	

Checked by	Signature
Date	

6. Project Database and Post-Implementation Review

This section deals with tuning the estimation tool to organisation. At the completion of each project, relevant information should be entered into the databases, and a post implementation review (PIR) conducted. This adjusts a number of values used throughout the estimation process, which will tailor the tool for the organisation, and increase the accuracy of subsequent estimates. As previously mentioned, the feature points and COCOMO processes depend heavily on past data being available and so the PIR plays a critical part in the estimation process. Don't leave it out!

6.1 Feature Point Database and PIR

This section develops a function to convert the number of feature points into a value for effort.

The following information will be needed for each project:

- Total Actual Effort (person hours)
- Final Feature Point Count

The information should be entered into the second and third columns of Figure 6.1. The first column should contain the name (or identifier) of the project.

Project Name	**Total Actual Effort in Person Hours (E)**	**Final Feature Point Count (C)**	**Actual Hours per Feature Point (R) = (E)/(C)**	**Predicted value of Hours per Feature Point (R')**	**Magnitude of Relative Error of R (MRE)**
				MMRE	

Figure 6.1

Next, for each project calculate the number of hours per feature point (R) , given by:

$$R = E / C$$

And enter the result into column four.

The value of R is a function of C. ie.

$$R(C) = f(C)$$

Solve for f(C) by plotting final hours per feature point (R), against final feature point count (C), from the above table and drawing a best fit graph or by using a suitable software package. We recommend an exponential model of the form $f(C) = ae^{kC}$ where *a* and *k* are constants, or a power function of the form $f(C) = aC^{b}$ where *a* and *b* are constants.

Now that the function f(C) has been found, calculate the value of R` for each project. In the case where more than one project exists in the database, the value of R` must be re-calculated using the latest equation for f(C) for each project.

Calculate the Magnitude of Relative Error of R in the above table using the following equation:

$$MRE = \frac{|R' - R|}{R}$$

To calculate the Mean Magnitude of Relative Error, simply take the mean of the MREs for the previous projects:

$$MMRE = \frac{1}{n}\sum_{i=1}^{n} MRE_i$$

Where n is the number of previous projects, and MRE_i is the Magnitude or Relative Error for the i^{th} project.

The value of MMRE is used in the feature point estimation, to provide a confidence interval.

6.2 COCOMO II Database and PIR

The COCOMO II PIR consists of two parts:

- Revise the feature point to KSLOC factors
- Revise the value of A

The first part will only have to be performed for projects that estimated size from the number of feature points. If the size for the COCOMO model was estimated directly, proceed to the second part of the COCOMO PIR.

6.2.1 Feature Point to KSLOC factor

The following information will be required:

- Final Feature Point count (C)
- Actual size in KSLOC (S).

⚠ It is imperative that the standards agreed for counting lines of code are adhered to. A checklist similar to that in Appendix B should be used as a guide on how to count lines of code.

Enter the values for C and S into Figure 6.2.

Project	**Language**	**Final Feature Point Count (C)**	**Actual KSLOC (S)**	**KSLOC per FP (F)**
			Average	

Figure 6.2

To find the feature point to KSLOC factor (F) for a specific language, calculate the value for the number of kilo lines of code per feature point using the formula given below and enter the result into column five.

$$F = S / C$$

Take the average of the values in the fifth column (only for projects that used the required language). This can be entered into Figure 2.6, and used as the new conversion factor.

If the values of S / C show a large variation, consider graphing the values of C and S / C, and fitting a function to the graph. This will result in an equation of the form:

$$F(C) = f(C)$$

Where f(C) is an appropriate function (linear, exponential etc.). This is similar to the method used for the Feature Point PIR (section 6.1)

6.2.2 Value of A

The following information will be required:

- Actual size in KSLOC (S)
- The β value used in the estimation
- The M value used in the estimation
- Total Actual Effort (person months)

Enter the values for S, β and M into Figure 6.3.

Next calculated the value of Ê, using the following formula:

$$\hat{E} = S^{\beta} \times M$$

Calculate the value for A' for each project, using the following formula:

$$A' = E / E'$$

Finally, average to values of A', to produce a new value A. This value can then be used in subsequent estimates. It should lie somewhere between 2.5 and 3.0.

Project	Actual KSLOC (S)	Estimated β	Estimated M	$\hat{E}$	Actual Effort (E)	A’
					Average (A)	

Figure 6.3

6.3 PERT Database and PIR

As each Estimable Work Unit is finished, during the development of the project, each estimator should record the actual time for completion. This is written in their own, personal *Estimator History Form*. From a comparison between their own estimates, and the actual durations, their Estimator Confidence value can be calculated. A description of each field in the *Estimator History Form* is below:

- Estimator Name

  The name of the estimator, who's estimates will be recorded on the same form.
- Work Unit Name

  Name of the work unit, which will be described on the same row.
- Unadjusted, Weighted Estimate (E_u)

  The estimator's Unadjusted, Weighted Estimate for the work unit given.
- Actual Duration (A)

  The actual time it took for the given work unit to complete.
- Difference (D)

  The difference between the estimator's time and the actual time. Calculated with:

  $$D = \frac{A - E_u}{A}$$
- Estimator Confidence (K)

  The final calculated value for the Estimator Confidence. It is basically based on the average of all the differences. Find it with the following formula:

  $$K = 1 + \frac{\sum D}{n}$$

Estimator Name	

Work Unit Name	Unadjusted, Weighted Estimate (E_u)	Actual Duration (A)	Difference (D)

Figure 6.4

Estimator Confidence (K)	

6.4 Process Combination Database and PIR

This section adjusts the weights for the individual models in the final combination formula.

The following information is required:

- Actual duration (D)
- Expected duration from Feature Points (E1)
- Expected duration from COCOMO (E2)
- Expected duration from PERT (E3)

Enter the values into Figure 6.5, along with the project name (or identifier).

Compute the accuracy of each process using the following formula:

$$\text{AW} = \frac{D}{|E1 - D|}$$

and enter the values into Figure 6.5.

Finally calculate the geometric mean of the accuracies for each process using the following formula:

$$\text{Geometric Mean} = G_n = \left(\prod_{i=1}^{n} x_i\right)^{\frac{1}{n}}$$

Where the values for x_i are the individual cells in a particular column while n is the number of values averaged. The final values for each weight can then be used in subsequent estimations.

Project No.	Actual Duration (D)	Feature Point Analysis Estimation (Expected) [E1]	COCOMO II Estimation (Expected) [E2]	PERT Estimation (Expected) [E3]	Accuracy weight of E1 = [AW1] = $\frac{D}{\lvert E1-D\rvert}$	Accuracy weight of E2 = [AW2] = $\frac{D}{\lvert E2-D\rvert}$	Accuracy weight of E3 = [AW3] = $\frac{D}{\lvert E3-D\rvert}$
					Geometric Mean:			

Figure 6.5

7.0 Estimation Example

7.1 Problem Statement

This section demonstrates the use of the tool to estimate a simple software project. The project under question formed part of the assessment for the Algorithms 300 Course, during semester 1, 2004 at the University of Western Australia. A short description of the project and relevant details is provided below.

The goal of the project was to provide an implementation of the Edmonds-Karp algorithm, which would calculate the maximum flow through a network. The network consisted of vertices (nodes) and edges as shown in Figure 7.1.

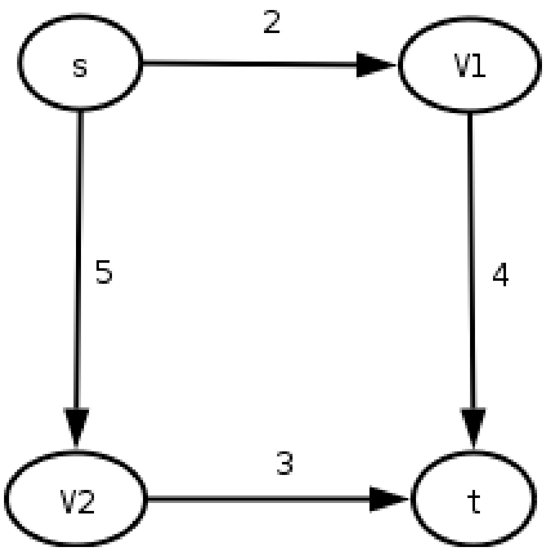


Figure 7.1

Each edge has a specific capacity (shown on the diagram above). Using the analogy of a water network, the edges represent pipes, and the aim of the algorithm is to determine the maximum amount of water (flow) that can be moved between the vertex marked s (the source) and the vertex marked t (the sink). For more information on the Edmonds-Karp algorithm, consult a book such as *Introduction to Algorithms* (Corman et al. 2001)

Data representing the network was supplied in a file in a form similar to that shown below:

```
4
0 2 5 0
0 0 0 4
0 0 0 3
0 0 0 0
```

The program was to have a graphical user interface, which would allow the user to load a data file, view the network on screen, and then be able to run the algorithm and view the results. In addition, a command line version was developed which simply printed the maximum flow.

7.2 Feature Points Example

7.2.1 Feature Point Count

The following feature points where identified:

Algorithms:
- Parsing
- Normalising the network
- Breath first search
- Finding the augmenting path
- Adjusting the flows and calculating residuals

External Inputs
- The data file

External Outputs
- Command line output
- GUI dialog
- GUI display (vertices and edges)

External Inquiries
(none)

Internal Logic files
- Vertex
- Edge
- Adjacency List
- Graph
- Queue

External Interface files
(none)

Next the complexity of each feature point was considered. All was ranted low except the graph normalisation and the GUI display, which were rated average. This data was entered into Figure 7.2 and gave an unadjusted feature point count of 54.

Component	Complexity			
	Low	Average	High	Total
Algorithms	__4_ × 2 = __8_	__1_ × 3 = __3_	__0_ × 5 = __0_	11
External Inputs	__1_ × 3 = __3_	__0_ × 4 = __0_	__0_ × 6 = __0_	3
External Outputs	__2_ × 4 = __8_	__1_ × 5 = __5_	__0_ × 7 = __0_	13
External Inquiries	__0_ × 3 = __0_	__0_ × 4 = __0_	__0_ × 6 = __0_	0
Internal Logic Files	__4_ × 5 = _20_	__1_ × 7 = __7_	__0_ × 10 = __0_	27
External Interface Files	__0_ × 5 = __0_	__0_ × 7 = __0_	__0_ × 10 = __0_	0
			Total Unadjusted Feature Points [A1]	54
			Complexity Multiplier [A2]	1.0
			Total Adjusted Feature Points [A3]	54

Figure 7.2

7.2.2 Complexity Multiplier

Problem Complexity: __3__

1) Simple algorithms and simple calculations
2) Majority of simple algorithms and calculations
3) Algorithms and calculations of average complexity
4) Some difficult or complex algorithms
5) Many difficult algorithms and complex calculations

Data Complexity: __3___

1) Simple data with few variables
2) Numerous variables, but simple data relationships
3) Multiple files, fields, and data intersections
4) Complex file structures and data intersections
5) Very complex file structures and data intersections

Sum of Problem and Data Complexity	2	3	4	5	**6**	7	8	9	10
Complexity Multiplier	0.6	0.7	0.8	0.9	**1.0**	1.1	1.2	1.3	1.4

Figure 7.3

Complexity Multiplier [A2] = __1.0___

Estimator's Comments

The problem complexity was rated as 3 (average) and the data complexity was also rated as 3 (due to the use of queues, lists and vectors).

This gave a sum or 6, which translates to a complexity multiplier of 1.0

The final adjusted feature point count is then 54.

7.2.3 Effort

Hours Per Month (HPM) = __85_______ (Default is 85)

MMRE = ___0________

R(C) = ___1.41 e $^{0.0003049\ C}$________________________ (Equation)

Effort $_{\text{optimistic}}$ $= [A3] \times 0.88 \times (1 - MMRE) \times R([A3] \times 0.88) \times \frac{1}{HPM}$

= __0.80_________ Person Months

Effort $_{\text{expected}}$ $= [A3] \times R([A3]) \times \frac{1}{HPM}$

= __0.91________ Person Months

Effort $_{\text{pessimistic}}$ $= [A3] \times 1.12 \times (1 + MMRE) \times R([A3] \times 1.12) \times \frac{1}{HPM}$

= __1.02_________ Person Months

Note: Refer to the PIR section 6.1 for the function of R, and value of MMRE

7.2.4 Duration

Duration = **a** Total Effort $^{\mathbf{b}}$

Duration $_{\text{optimistic}}$ = **a** (Effort $_{\text{optimistic}}$)$^{\mathbf{b}}$ = __2.3_____ Months

Duration $_{\text{expected}}$ = **a** (Effort $_{\text{expected}}$)$^{\mathbf{b}}$ = __2.4_____ Months

Duration $_{\text{pessimistic}}$ = **a** (Effort $_{\text{pessimistic}}$)$^{\mathbf{b}}$ = __2.5_____ Months

Note: See section 2.1.5 for selecting appropriate values of **a** and **b**

Estimator's Comments:

Since no past data was available, we used the supplied formula for R(C) = 1.41 e $^{0.0003049\ C}$. As noted in the user guide, the lack of past data restricts the value of the confidence interval.

The system was determined to be organic, which gave a= 2.5 and b = 0.38.

7.3 COCOMO II - Templates

7.3.1 Estimating Size

⚠ To calculate system size, either use unadjusted function points (section 2.2.1.1) , **OR** estimate the lines of code on a per module basis (section 2.2.1.2)

Using Unadjusted Feature Points

Unadjusted function point count from Function Point Analysis = ___54________ [B1]

Equivalent SLOC = ___1242________________ [B2]

Size of System in KSLOC : ([B2] / 1000) = _______1.242__________________ [B3]

Estimator's Comments:

In this project we decided to estimate size from feature points, rather than perform direct estimation for the number of lines of code. As it happened, the estimated value was 1.242 KSLOC which was remarkably similar to the actual value for the project of 1.228 KSLOC. This completely changed the group's faith in estimation tools!

7.3.2 Scale Factors

Factor	**Value**	
PREC	2	
FLEX	1	
RESL	5	
TEAM	2	
PMAT	4	
Sum	**14**	**[B4]**

Figure 7.4

β = 1.01 + 0.01 x [B4]

Value of exponent β = ____1.14____________ [B5]

Estimator's Comments:

PREC – this type of project was generally familiar to the group. All members has past experience in Java and two of the three members had experience in Java GUIs.

FLEX – The team had a fair degree of flexibility in the development process. The main objectives were specified, but the team had to have bi-weekly meetings and complete minute sheets.

RESL – no risk resolution was in place.

TEAM – Interactions in the team were largely co-operative. Although this was a new team which had no previous experience working together.

PMAT – No software improvement process was in place.

7.3.3 Cost Drivers

Driver	**Value**
RELY	0.88
DATA	0.93
DOCU	1
RUSE	1
TIME	1
STOR	1
PVOL	0.87
ACAP	1
PCAP	0.87
PCON	1
AEXP	1
PEXP	0.88
LTEX	0.91
TOOL	1.2
SITE	0.85
CPLX	0.952
Product [B6]	**0.4816**

Figure 7.5

Estimator's Comments:

RELY – The software was not mission critical, errors while no desired did not pose a large risk
DATA – The size of the data files was estimated to be at most a few hundred kilobytes. This meant the D/P factor was less than 1.
DOCU – The amount of documentation required was the right size for the project. All methods were required to have Javadoc, and a short user manual was to be produced.
RUSE – while components were not expected to be reused, interfaces between modules were well defined and the application design was modular.
TIME – execution time was not an issue
STOR – storage was not an issue
PVOL – the platform was well defined (Java 1.4) and stable
ACAP – the analyst capability was estimated to be in the 55 to 75th percentile.
PCAP – the programmer capability was estimated to be in the 75th percentile
PCON – personnel turnover was not a factor (set to 1)
AEXP – application experience was approximately 1 year
PXEP – platform experience was between 2 and 3 years
LTEX – language and tool experience was also between 2 and 3 years
TOOL – No CASE tools were used.
SITE – The project team were in the same building for the majority of the project.
CPLX – The average of the CMPX factors came to 0.952

Control – Nominal
Computational – Nominal
Device dependant – Low
Data Management – Low
UI Management – Nominal

7.3.4 Expected Effort

Value of A = _3.0_____________ (from PIR or default to 3.0)

$$\text{Effort} = E = A \times [B3]^{[B5]} \times [B6]$$

Effort (Person Months) Estimate = _1.84___________ [B7]

7.3.5 Confidence Interval

Required Confidence Level γ = ____0.95____

Area under tail of normal curve: $\alpha = (1-\gamma)/2$ = __0.025_______

Lower cutoff z_L = ____- 1.96_______

Upper cutoff z_H = _____1.96________

Optimistic Effort E_O = ___0.65____ [B8]

Pessimistic Effort E_P = ____3.66___ [B9]

7.3.6 Duration

$$\text{Duration} = A \times E^{0.33 + 0.2 \times ([B5] - 1.01)}$$

Where E is either [B8], [B7] or [B9] for optimistic, expected and pessimistic duration respectively.

Optimistic Duration Estimate: ___2.57___ __________ (months)

Expected Duration Estimate: ____3.73 _____________ (months)

Pessimistic Duration Estimate: ___4.78_____________ (months)

Estimator's Comments:

As no past data was available, the value of A was set to 3.0

A 95% confidence interval was selected.

The estimation was done at the end of the requirements analysis stage (but before detailed design commenced)

7.4 PERT Estimation Example

The project was divided into 6 work units:

- Base Classes
  These were the basic classes used to represent the objects required by the algorithm. The initial classes identified were; Edge, Vertex, AdjacencyList and Graph

- Parser
  This would parser the data file into the data structures required by the rest of the program

- Breath First Search (BFS)
  This was the algorithm that would perform a breath first search on the residual network

- Edmonds Karp
  This was the main algorithm, responsible for solving the maximum flow problem

- GUI
  This was the user interface code.

- User Manual
  This was the manual to describe how to use the system, and also explain the output from some trial runs.

This example shows forms completed by two estimators for the work unit classes “Base Classes”. The other forms for the other units have been omitted for clarity, although the procedure is identical.

Due to the small size of the project, the durations were estimated in hours and converted to days, using the ratio of 5 working hours per day.

7.4.1 Individual Estimator Forms

Work Unit Name	Base Classes
Estimator Name	SN

Team Size	3

Estimates (Duration in Days)				
Optimistic (O)		Realistic (R)		Pessimistic (P)
0.4		1		3

Figure 7.6

Work Unit Name	Base Classes
Estimator Name	PM

Team Size	3

Estimates (Duration in Days)				
Optimistic (O)		Realistic (R)		Pessimistic (P)
2		3		5

Figure 7.7

7.4.2 Work Unit Estimate Form

Work Unit Name	Base Classes

Estimator Name	Optimistic (O)	Realistic (R)	Pessimistic (P)	Unadjusted, Weighted Estimate (E_u)	Variance (σ_u^2)	Estimator Confidence (K)	Adjusted Estimate (E_a)	Adjusted Variance (σ_a^2)
SN	0.4	1	3	1.07	0.187	1	1.07	0.187
PM	2	3	5	3.1	0.25	1	3.1	0.25

Figure 7.8

Average, Adjusted Estimate (E_v)	2.1
Average, Adjusted Variance (σ_v^2)	0.22

7.4.3 Project Estimate Form

Work Unit Name	Average, Adjusted Estimate (E_v)	Average, Adjusted Variance (σ_v^2)
Base Classes	2.1	0.22
Parser	0.7	0.19
BFS	1.2	0.37
Edmonds Karp	2.3	0.43
GUI	4.1	0.78
User Manual	1.8	0.11

Figure 7.9

Total Project Estimate (E)	12.2
Total Project Variance (σ^2)	2.1
Estimate Standard Deviation (σ)	1.45

Duration $_{\text{optimistic}} = E - 2\sigma$ = ___9.3________ Days = __0.55___ Months

Duration $_{\text{expected}} = E$ = ___12.2_______ Days = __0.72___ Months

Duration $_{\text{pessimistic}} = E + 2\sigma$ = ___15.1_______ Days = __0.88___ Months

Effort in person days was converted to person months using the ratio of 17 working days per month.

7.5 Process Combination Templates

Revision Number	1
Performed By	SN, PM, SW
on	

Method	**Optimistic Estimation of Duration**	**Expected Duration**	**Pessimistic Estimation of Duration**	**Accuracy Weight**
Feature Point Analysis	[O1] = 2.3	[E1] = 2.4	[P1] = 2.5	[AW1] = 1.0
COCOMO II Analysis	[O2] = 2.57	[E2] = 3.73	[P2] = 4.78	[AW2] = 1.0
PERT Analysis	[O3] = 0.55	[E3] = 0.72	[P3] = 0.88	[AW3] = 1.0

Figure 7.10

O = **Final Optimistic Duration** = $\frac{(O1 \times AW1) + (O2 \times AW2) + (O3 \times AW3)}{AW1 + AW2 + AW3}$ = _1.80___ Months

E = **Final Expected Duration** = $\frac{(E1 \times AW1) + (E2 \times AW2) + (E3 \times AW3)}{AW1 + AW2 + AW3}$ = _2.28___ Months

P = **Final Pessimistic Duration** = $\frac{(P1 \times AW1) + (P2 \times AW2) + (P3 \times AW3)}{AW1 + AW2 + AW3}$ = _2.72___ Months

ERE = **Expected Risk Exposure** = ______________________________ (person –days)

Next revision due in ______1________ months

Estimator's Comments:

Risk analysis was not performed in this example.

Interestingly enough the total time taken to complete the project was 54 hours, 10.8 person days or 0.63 person months. This did not include time for group meetings which would have added another 27 hours to the project.

Obviously there is a discrepancy between the three models used. Being a small project, the numbers used in the feature point and COCOMO models were much lower than the values that would typically be encountered in reasonably sized projects. It is quite likely that the models are not valid for such small numbers, and in this case the PERT estimate should been given a higher rating. This would take place in the PIR section.

8. Glossary

COCOMO – Constructive Cost Model. A method for evaluating the effort required to develop software given an estimate of the software's size.

ERE – Expected Risk Exposure. An indication of the amount of additional effort the project will require, if the expected number of risks occur.

HPM – Hours per month. Highly orgnaisational dependant, in this tool a default values of 85 hours per months has been used (5 hours per day, 17 days per month)

KSLOC – Kilo Source Lines of Code. See SLOC

MMRE – Mean Magnitude of Relative Error. The average of the MRE for all data points

MRE – Magnitude of relative error. The variation between the estimated value and the actual value for an individual data point.

PERT – Program Evaluation and Review Technique. An estimation technique based on individual estimates from a group of people. The differences between each individual's estimate can also be used to calculate the standard deviation.

PIR – Post Implementation Review. A procedure that is performed at the end of the project, to tune the parameters used by the estimation tool and increase the accuracy for future estimations.

RE – Risk Exposure. A measure of the severity of a risk. Found from multiplier the probability the risk occurs by the impact in the event that the risk occurs.

Revision - One iteration of the estimation tool. It is expected that many revisions (estimates) will be prepared for each project. See section 4.0 on what events should trigger a revision.

SLOC – Source Lines of Code. Use to describe the number of lines of code contained within a program

9. Bibliography

Angelis, L & Stamelos, I. 2000, *A Simulation Tool for Efficient Analogy Based Cost Estimation*, Empirical Software Engineering, vol. 5, no. 1, pp. 35-68.

Boehm, B. 1981, *Software Engineering Economics*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey.

Boehm, B., 1991, *Software risk management: principles and practices*, IEEE Software, vol. 8, no. 1, pp. 32-41.

Boehm, B, et al., 1995*, Cost Models for Future Software Life Cycle Processes: COCOMO 2.0,* Annals of Software Engineering 1, pp 1-24.

Boehm, B. 2000, *COCOMO II Model Definition Manual*, [Online], Available from: <http://sunset.usc.edu/research/COCOMOII/Docs/modelman.pdf> [1 Apr 2004]

Cormen, T. et al., 2001, *Introduction to Algorithms*, 2nd edn, The MIT Press.

Douglass, B. P. 1996 - 2004, *Software Estimation and Scheduling*, [Online], TechOnLine, Available from: <http://www.techonline.com/community/tech_group/embedded/tech_paper/5947/>, [20 April 2004].

Hotowitz, E. 1998, *USC COCOMO II User Manual*, [Online], Available from: <http://sunset.usc.edu/research/COCOMOII/Docs/usersman.pdf>, [6 April2004]

Johnson, K, Jan 1998, *Software Size Estimation*, [Online], Department of Computer Science, University of Calgary, Available from: <http://sern.ucalgary.ca/courses/seng/621/W98/johnsonk/software.htm>, [8 April 2004].

Kemerer, C. 1993, *Reliability of function points measurement: a Field Experiment*, Communications of the ACM, vol 36, no. 2, pp 85 – 97.

Kitchenham, B. & Linkman, S 1997, *Estimates, Uncertainty and Risk*, IEEE Software, vol. 14, no. 3, pp 69-74.

Leon-Garcia, A. 1994, *Probability and Random Processes for Electrical Engineering*, 2nd end, Addison Wesley, USA

Longstreet Consulting Inc, 2003, *Improved Function Point Definitions*, [Online], Longstreet Consulting Inc., Available from <http://www.ifpug.com/trainingcourse/definitions.htm> [29 March 2004]

Longstreet, D., 2003, *Estimating Software Development Effort Using Function Points*, [Online], Longstreet Consulting Inc., Available from <http://www.ifpug.com/Articles/estimatingdata.htm> [28 March 2004]

Longstreet, D., 2003, *Fundamentals of Function Point Analysis*, [Online], Longstreet Consulting Inc., Available from <http://www.ifpug.com/fpafund.htm> [28 March 2004]

Miyazaki, Y. & Kuniaki M. 1985, *COCOMO Evaluation and Tailoring*, IEEE Computer Society Press [14 April 2004]

Park, R. 1992, *Software Size Measurement: A Framework for Counting Source Statements,* Software Engineering Institute, Ch 23.

Paulk, M., Curtis, B., 1993, *Capability and Maturity Model, Version 1.1*, IEEE Software, vol. 10, no. 4, pp 68-70.

Pressman, R.S.1997, *Software Engineering – A Practitioner's Approach*, 4th edn, McGraw Hill, USA

Robare, B. & Short, D., 3 August 99, *Function Points and SLOC,* [Online], Naval Postgraduate School (NPS), Available from <http://www.nps.navy.mil/wings/acq_topics/synopsis/archives/Summer%2099/fpbrief.ppt> [6 April 2004]

Sommerville, I. 2001, *Software Engineering*, 6th edn, Addison Wesley, Essex

What Are Feature Points, 6 August 2002, *What Are Feature Points?*, [Online], Software Productivity Research, Available from <http://www.spr.com/products/feature.htm> [26 March 2004]

Williams, T. M., 1996, *The two-dimensionality of Project Risk*, International Journal of Project Management, vol. 14, no. 3, pp. 185-186.

Appendix A – Cumulative Probabilities for the Normal Distribution

P(Z ≤ z) where Z ~ N(0,1)

z	0	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09
-3	0.0013	0.0013	0.0013	0.0013	0.0013	0.0013	0.0013	0.0013	0.0013	0.0013
-2.9	0.0018	0.0018	0.0018	0.0018	0.0018	0.0018	0.0018	0.0018	0.0019	0.0019
-2.8	0.0025	0.0025	0.0025	0.0025	0.0025	0.0025	0.0025	0.0025	0.0026	0.0026
-2.7	0.0034	0.0034	0.0034	0.0034	0.0034	0.0034	0.0034	0.0034	0.0035	0.0035
-2.6	0.0046	0.0046	0.0046	0.0046	0.0046	0.0046	0.0046	0.0046	0.0047	0.0047
-2.5	0.0061	0.0061	0.0061	0.0061	0.0061	0.0061	0.0061	0.0061	0.0062	0.0062
-2.4	0.0081	0.0081	0.0081	0.0081	0.0081	0.0081	0.0081	0.0081	0.0082	0.0082
-2.3	0.0106	0.0106	0.0106	0.0106	0.0106	0.0106	0.0106	0.0106	0.0107	0.0107
-2.2	0.0138	0.0138	0.0138	0.0138	0.0138	0.0138	0.0138	0.0138	0.0139	0.0139
-2.1	0.0177	0.0177	0.0177	0.0177	0.0177	0.0177	0.0177	0.0177	0.0179	0.0179
-2	0.0226	0.0226	0.0226	0.0226	0.0226	0.0226	0.0226	0.0226	0.0228	0.0228
-1.9	0.0285	0.0285	0.0285	0.0285	0.0285	0.0285	0.0285	0.0285	0.0287	0.0287
-1.8	0.0357	0.0357	0.0357	0.0357	0.0357	0.0357	0.0357	0.0357	0.0359	0.0359
-1.7	0.0442	0.0442	0.0442	0.0442	0.0442	0.0442	0.0442	0.0442	0.0446	0.0446
-1.6	0.0544	0.0544	0.0544	0.0544	0.0544	0.0544	0.0544	0.0544	0.0548	0.0548
-1.5	0.0664	0.0664	0.0664	0.0664	0.0664	0.0664	0.0664	0.0664	0.0668	0.0668
-1.4	0.0802	0.0802	0.0802	0.0802	0.0802	0.0802	0.0802	0.0802	0.0808	0.0808
-1.3	0.0962	0.0962	0.0962	0.0962	0.0962	0.0962	0.0962	0.0962	0.0968	0.0968
-1.2	0.1144	0.1144	0.1144	0.1144	0.1144	0.1144	0.1144	0.1144	0.1151	0.1151
-1.1	0.1349	0.1349	0.1349	0.1349	0.1349	0.1349	0.1349	0.1349	0.1357	0.1357
-1	0.1578	0.1578	0.1578	0.1578	0.1578	0.1578	0.1578	0.1578	0.1587	0.1587
-0.9	0.1831	0.1831	0.1831	0.1831	0.1831	0.1831	0.1831	0.1831	0.1841	0.1841
-0.8	0.2109	0.2109	0.2109	0.2109	0.2109	0.2109	0.2109	0.2109	0.2119	0.2119
-0.7	0.2409	0.2409	0.2409	0.2409	0.2409	0.2409	0.2409	0.2409	0.2420	0.2420
-0.6	0.2731	0.2731	0.2731	0.2731	0.2731	0.2731	0.2731	0.2731	0.2743	0.2743
-0.5	0.3073	0.3073	0.3073	0.3073	0.3073	0.3073	0.3073	0.3073	0.3085	0.3085
-0.4	0.3433	0.3433	0.3433	0.3433	0.3433	0.3433	0.3433	0.3433	0.3446	0.3446
-0.3	0.3808	0.3808	0.3808	0.3808	0.3808	0.3808	0.3808	0.3808	0.3821	0.3821
-0.2	0.4194	0.4194	0.4194	0.4194	0.4194	0.4194	0.4194	0.4194	0.4207	0.4207
-0.1	0.4588	0.4588	0.4588	0.4588	0.4588	0.4588	0.4588	0.4588	0.4602	0.4602
0	0.4986	0.4986	0.4986	0.4986	0.4986	0.4986	0.4986	0.4986	0.5000	0.5000
0	0.5014	0.5014	0.5014	0.5014	0.5014	0.5014	0.5014	0.5014	0.5000	0.5000
0.1	0.5412	0.5412	0.5412	0.5412	0.5412	0.5412	0.5412	0.5412	0.5398	0.5398
0.2	0.5806	0.5806	0.5806	0.5806	0.5806	0.5806	0.5806	0.5806	0.5793	0.5793
0.3	0.6192	0.6192	0.6192	0.6192	0.6192	0.6192	0.6192	0.6192	0.6179	0.6179
0.4	0.6567	0.6567	0.6567	0.6567	0.6567	0.6567	0.6567	0.6567	0.6554	0.6554
0.5	0.6927	0.6927	0.6927	0.6927	0.6927	0.6927	0.6927	0.6927	0.6915	0.6915
0.6	0.7269	0.7269	0.7269	0.7269	0.7269	0.7269	0.7269	0.7269	0.7257	0.7257
0.7	0.7591	0.7591	0.7591	0.7591	0.7591	0.7591	0.7591	0.7591	0.7580	0.7580
0.8	0.7891	0.7891	0.7891	0.7891	0.7891	0.7891	0.7891	0.7891	0.7881	0.7881
0.9	0.8169	0.8169	0.8169	0.8169	0.8169	0.8169	0.8169	0.8169	0.8159	0.8159
1	0.8422	0.8422	0.8422	0.8422	0.8422	0.8422	0.8422	0.8422	0.8413	0.8413
1.1	0.8651	0.8651	0.8651	0.8651	0.8651	0.8651	0.8651	0.8651	0.8643	0.8643
1.2	0.8856	0.8856	0.8856	0.8856	0.8856	0.8856	0.8856	0.8856	0.8849	0.8849
1.3	0.9038	0.9038	0.9038	0.9038	0.9038	0.9038	0.9038	0.9038	0.9032	0.9032
1.4	0.9198	0.9198	0.9198	0.9198	0.9198	0.9198	0.9198	0.9198	0.9192	0.9192
1.5	0.9336	0.9336	0.9336	0.9336	0.9336	0.9336	0.9336	0.9336	0.9332	0.9332
1.6	0.9456	0.9456	0.9456	0.9456	0.9456	0.9456	0.9456	0.9456	0.9452	0.9452
1.7	0.9558	0.9558	0.9558	0.9558	0.9558	0.9558	0.9558	0.9558	0.9554	0.9554
1.8	0.9643	0.9643	0.9643	0.9643	0.9643	0.9643	0.9643	0.9643	0.9641	0.9641
1.9	0.9715	0.9715	0.9715	0.9715	0.9715	0.9715	0.9715	0.9715	0.9713	0.9713
2	0.9774	0.9774	0.9774	0.9774	0.9774	0.9774	0.9774	0.9774	0.9772	0.9772
2.1	0.9823	0.9823	0.9823	0.9823	0.9823	0.9823	0.9823	0.9823	0.9821	0.9821
2.2	0.9862	0.9862	0.9862	0.9862	0.9862	0.9862	0.9862	0.9862	0.9861	0.9861
2.3	0.9894	0.9894	0.9894	0.9894	0.9894	0.9894	0.9894	0.9894	0.9893	0.9893
2.4	0.9919	0.9919	0.9919	0.9919	0.9919	0.9919	0.9919	0.9919	0.9918	0.9918
2.5	0.9939	0.9939	0.9939	0.9939	0.9939	0.9939	0.9939	0.9939	0.9938	0.9938
2.6	0.9954	0.9954	0.9954	0.9954	0.9954	0.9954	0.9954	0.9954	0.9953	0.9953
2.7	0.9966	0.9966	0.9966	0.9966	0.9966	0.9966	0.9966	0.9966	0.9965	0.9965
2.8	0.9975	0.9975	0.9975	0.9975	0.9975	0.9975	0.9975	0.9975	0.9974	0.9974
2.9	0.9982	0.9982	0.9982	0.9982	0.9982	0.9982	0.9982	0.9982	0.9981	0.9981
3	0.9987	0.9987	0.9987	0.9987	0.9987	0.9987	0.9987	0.9987	0.9987	0.9987

Appendix B – Code Counting Checklist

This is an example checklist for counting lines of code, taken from *Software Size Measurement: A Framework for Counting Source Statements* (Park 1992). For each type of statement (executable, declarations, compiler directives etc.) a decision must be made as to whether statements of this type will be included in the final KSLOC count. Once a decision has been made, a tick is placed in either the *include* or *exclude* column. This process is then repeated for the rest of the template.

When the code is to be counted. The first category that the line matches, is the one that decided if it is included in the count or not. So for example, a line that has an executable statement, as well as a comment, would match the *Executable* statement type (and presumably be included in the count).

Space has been left throughout the template for the organisation to include it's own definitions

Definition Name		Date Originator	
Statement Type		**Include**	**Exclude**
1 Excecutable Order of Precedence ->	1		
2 Nonexecutable			
3 Declarations	2		
4 Compiler Directives	3		
5 Comments			
6 On their own lines	4		
7 On lines with source code	5		
8 Banners and non black spaces	6		
9 Blank (empty) comments	7		
10 Blank lines	8		
11			
12			
How Produced		**Include**	**Exclude**
1 Programmed			
2 Generated with source code generators			
3 Converted with automated translators			
4 Copied or re-used without change			
5 Modified			
6 Removed			
7			
8			
Origin		**Include**	**Exclude**
1 New work: no prior existance			
2 Prior work: taken or adapted from			
3 A previous version or build			
4 Commercial, off the shelf, other than libraries			
5 Government furnished software, other than reuse libraries			
6 Another product			
7 A vendor supplied language support library			
8 A vendor supplied operating system or utility			
9 A local or modified language library of operating system			
10 Other commercial library			
11 A reuse library (software designed for reuse)			
12 Other software component or library			
13			
14			
Usage		**Include**	**Exclude**
1 In or as part of the primary product			
2 External to or in support of the primary product			
3			

Appendix C – Example Code Counting Checklist

Definition Name Physical Source Lines of Code			**Date** 8/7/92 **Originator** SEI	
Statement Type			**Include**	**Exclude**
1 Excecutable	Order of Precedence ->	1	●	
2 Nonexecutable				
3 Declarations		2	●	
4 Compiler Directives		3	●	
5 Comments				
6 On their own lines		4		●
7 On lines with source code		5		●
8 Banners and non black spaces		6		●
9 Blank (empty) comments		7		●
10 Blank lines		8		●
11				
12				
How Produced			**Include**	**Exclude**
1 Programmed			●	
2 Generated with source code generators			●	
3 Converted with automated translators			●	
4 Copied or re-used without change			●	
5 Modified			●	
6 Removed				●
7				
8				
Origin			**Include**	**Exclude**
1 New work: no prior existance			●	
2 Prior work: taken or adapted from				
3 A previous version or build			●	
4 Commercial, off the shelf, other than libraries			●	
5 Government furnished software, other than reuse libraries			●	
6 Another product			●	
7 A vendor supplied language support library				●
8 A vendor supplied operating system or utility				●
9 A local or modified language library of operating system			●	
10 Other commercial library				●
11 A reuse library (software designed for reuse)			●	
12 Other software component or library			●	
13				
14				
Usage			**Include**	**Exclude**
1 In or as part of the primary product			●	
2 External to or in support of the primary product				●
3				