Chapter 5

Software Project Planning

Software Project Planning

- After the finalization of SRS, we would like to estimate size, cost and development time of the project. Also, in many cases, customer may like to know the cost and development time even prior to finalization of the SRS.
- In order to conduct a successful software project, we must understand:
  - Scope of work to be done
  - Software Project Planning
  - The risk to be incurred
  - The resources required
  - The task to be accomplished
  - The cost to be expended
  - The schedule to be followed

Software Project Planning

- Software planning begins before technical work starts, continues as the software evolves from concept to reality, and culminates only when the software is retired.

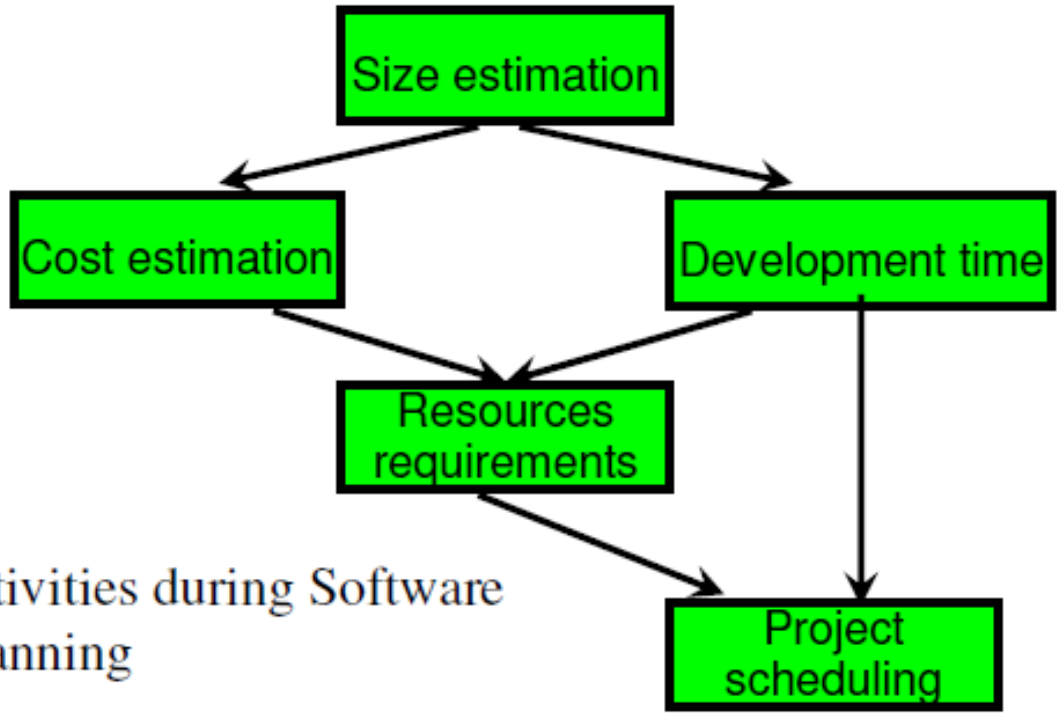


Fig. 1: Activities during Software Project Planning

Project size estimation techniques

- Estimation of the size of software is an essential part of Software Project Management. It helps the project manager to further predict the effort and time which will be needed to build the project. Various measures are used in project size estimation. Some of these are:
  - Lines of Code
  - Number of entities in ER diagram
  - Total number of processes in detailed data flow diagram
  - Function points

Lines of Code

- As the name suggest, LOC count the total number of lines of source code in a project. The units of LOC are:
  - KLOC- Thousand lines of code
  - NLOC- Non comment lines of code
  - KDSI- Thousands of delivered source instruction
- LOC is the simplest among all metrics available to estimate project size. This metric is very popular because it is the simplest to use. Using this metric, the project size is estimated by counting the number of source instructions in the developed program. Obviously, while counting the number of source instructions, lines used for commenting the code and the header lines should be ignored.

Lines of Code

- Determining the LOC count at the end of a project is a very simple job. However, accurate estimation of the LOC count at the beginning of a project is very difficult. In order to estimate the LOC count at the beginning of a project, project managers usually divide the problem into modules, and each module into sub-modules and so on, until the sizes of the different leaf-level modules can be approximately predicted. To be able to do this, past experience in developing similar products is helpful. By using the estimation of the lowest level modules, project managers arrive at the total size estimation.

Lines of Code

- **Advantages:**
  - Universally accepted and is used in many models like COCOMO.
  - Estimation is closer to developer's perspective.
  - Simple to use.
- **Disadvantages:**
  - Different programming languages contains different number of lines.
  - No proper industry standard exist for this technique.
  - It is difficult to estimate the size using this technique in early stages of project.

Function Point Analysis - History

- The concept of Function Points was introduced by Alan Albrecht of IBM in 1979.
- In 1984, Albrecht refined the method.
- The first Function Point Guidelines were published in 1984.
- The International Function Point Users Group (IFPUG) is a US-based worldwide organization of Function Point Analysis metric software users. The **International Function Point Users Group (IFPUG)** is a non-profit, member-governed organization founded in 1986. IFPUG owns Function Point Analysis (FPA) as defined in ISO standard 20296:2009 which specifies the definitions, rules and steps for applying the IFPUG's functional size measurement (FSM) method. IFPUG maintains the Function Point Counting Practices Manual (CPM).

Function Point Analysis - History

- CPM 2.0 was released in 1987, and since then there have been several iterations. CPM Release 4.3 was in 2010.
- The CPM Release 4.3.1 with incorporated ISO editorial revisions was in 2010. The ISO Standard (IFPUG FSM) - Functional Size Measurement that is a part of CPM 4.3.1 is a technique for measuring software in terms of the functionality it delivers. The CPM is an internationally approved standard under ISO/IEC 14143-1 Information Technology – Software Measurement.

Function Point Analysis

- The steps in function point analysis are:
  - Count the number of functions of each proposed type.
  - Compute the Unadjusted Function Points (UFP).
  - Find Total Degree of Influence (TDI).
  - Compute Value Adjustment Factor (VAF).
  - Find the Function Point Count (FPC).

Function Point Analysis

- **Count the number of functions of each proposed type:** Find the number of functions belonging to the following types:
  - External Inputs: Functions related to data entering the system.
  - External outputs: Functions related to data exiting the system.
  - External Inquiries: They leads to data retrieval from system but don't change the system.
  - Internal Files: Logical files maintained within the system. Log files are not included here.
  - External interface Files: These are logical files held by other systems which are used by the system being analyzed.

Function Point Analysis

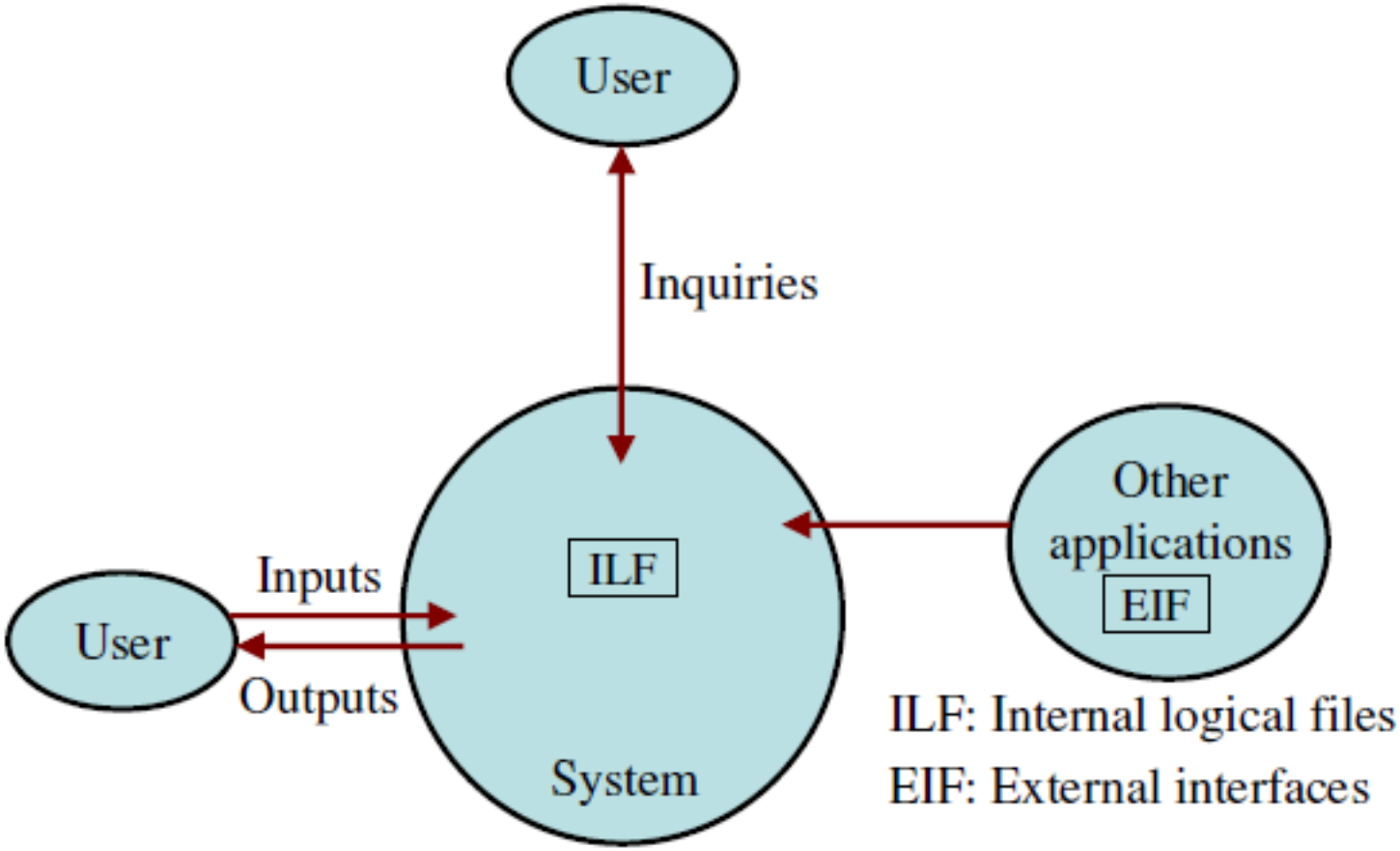


Fig. 3: FPAs functional units System

Function Point Analysis

- **Compute the Unadjusted Function Points (UFP):** Categorize each of the five function types as simple, average or complex based on their complexity. Multiply count of each function type with its weighting factor and find the weighted sum. The weighting factors for each type based on their complexity are as follows:

Functional Units	Weighting factors		
	Low	Average	High
External Inputs (EI)	3	4	6
External Output (EO)	4	5	7
External Inquiries (EQ)	3	4	6
External logical files (ILF)	7	10	15
External Interface files (EIF)	5	7	10

Table 1 : Functional units with weighting factors

Function Point Analysis

Table 2: UFP calculation table

Functional Units	Count Complexity		Complexity Totals	Functional Unit Totals
External Inputs (EIs)		Low x 3 =		
		Average x 4 =		
		High x 6 =		
External Outputs (EOs)		Low x 4 =		
		Average x 5 =		
		High x 7 =		
External Inquiries (EQs)		Low x 3 =		
		Average x 4 =		
		High x 6 =		
External logical Files (ILFs)		Low x 7 =		
		Average x 10 =		
		High x 15 =		
External Interface Files (EIFs)		Low x 5 =		
		Average x 7 =		
		High x 10 =		
Total Unadjusted Function Point Count				

Function Point Analysis

- The weighting factors are identified for all functional units and multiplied with the functional units accordingly. The procedure for the calculation of Unadjusted Function Point (UFP) is given in table shown above.

Function Point Analysis

- The procedure for the calculation of UFP in mathematical form is given below:

$$UFP = \sum_{i=1}^{5} \sum_{J=1}^{3} Z_{ij} w_{ij}$$

- Where i indicate the row and j indicates the column of Table 1
- W_{ij} : It is the entry of the i^{th} row and j^{th} column of the table 1
- Z_{ij} : It is the count of the number of functional units of Type *i* that have been classified as having the complexity corresponding to column *j*.

Function Point Analysis

- **Find Total Degree of Influence (TDI):** Use the ’14 general characteristics’ of a system to find the degree of influence of each of them. The sum of all 14 degrees of influences will give the TDI. The range of TDI is 0 to 70.
- **Compute Value Adjustment Factor (VAF):** Use the following formula to calculate VAF = (TDI * 0.01) + 0.65
- **Find the Function Point Count (FPC):** Use the following formula to calculate FPC = UFP * VAF

Function Point Analysis

1. Does the system require reliable backup and recovery ?
2. Is data communication required ?
3. Are there distributed processing functions ?
4. Is performance critical ?
5. Will the system run in an existing heavily utilized operational environment ?
6. Does the system require on line data entry ?
7. Does the on line data entry require the input transaction to be built over multiple screens or operations ?

Function Point Analysis

8. Are the master files updated on line ?

9. Is the inputs, outputs, files, or inquiries complex ?

10. Is the internal processing complex ?

11. Is the code designed to be reusable ?

12. Are conversion and installation included in the design ?

13. Is the system designed for multiple installations in different organizations ?

14. Is the application designed to facilitate change and ease of use by the user ?

Function Point Analysis

- Functions points may compute the following important metrics:
  - Productivity = FP / persons-months
  - Quality = Defects / FP
  - Cost = Rupees / FP
  - Documentation = Pages of documentation / FP
- These metrics are controversial and are not universally acceptable.

Function Point Analysis

Consider a project with the following functional units:

Number of user inputs = 50

Number of user outputs = 40

Number of user enquiries = 35

Number of user files = 06

Number of external interfaces = 04

Assume all complexity adjustment factors and weighting factors are average. Compute the function points for the project.

Function Point Analysis

Table 2: UFP calculation table

Functional Units	Count Complexity			Complexity Totals	Functional Unit Totals
External Inputs (EIs)		Low x 3	=		
		Average x 4	=		
		High x 6	=		
External Outputs (EOs)		Low x 4	=		
		Average x 5	=		
		High x 7	=		
External Inquiries (EQs)		Low x 3	=		
		Average x 4	=		
		High x 6	=		
External logical Files (ILFs)		Low x 7	=		
		Average x 10	=		
		High x 15	=		
External Interface Files (EIFs)		Low x 5	=		
		Average x 7	=		
		High x 10	=		
Total Unadjusted Function Point Count					

Function Point Analysis

Solution

We know

$$UFP = \sum_{i=1}^{5}\sum_{J=1}^{3} Z_{ij} w_{ij}$$

UFP = 50 x 4 + 40 x 5 + 35 x 4 + 6 x 10 + 4 x 7

= 200 + 200 + 140 + 60 + 28 = 628

CAF = $(0.65 + 0.01\ \Sigma F_i)$

= (0.65 + 0.01 (14 x 3)) = 0.65 + 0.42 = 1.07

FP = UFP x CAF

= 628 x 1.07 = 672

Function Point Analysis

An application has the following:

10 low external inputs, 12 high external outputs, 20 low internal logical files, 15 high external interface files, 12 average external inquiries, and a value of complexity adjustment factor of 1.10.

What are the unadjusted and adjusted function point counts ?

Function Point Analysis

Solution

Unadjusted function point counts may be calculated using as:

$$UFP = \sum_{i=1}^{5} \sum_{J=1}^{3} Z_{ij} w_{ij}$$

= 10 x 3 + 12 x 7 + 20 x 7 + 15 + 10 + 12 x 4

= 30 + 84 +140 + 150 + 48

= 452

FP = UFP x CAF

= 452 x 1.10 = 497.2.

Function Point Analysis

Consider a project with the following parameters.

(*i*) External Inputs:
- (a) 10 with low complexity
- (b) 15 with average complexity
- (c) 17 with high complexity

(*ii*) External Outputs:
- (a) 6 with low complexity
- (b) 13 with high complexity

(*iii*) External Inquiries:
- (a) 3 with low complexity
- (b) 4 with average complexity
- (c) 2 high complexity

Function Point Analysis

(*iv*) Internal logical files:

(a) 2 with average complexity

(b) 1 with high complexity

(v) External Interface files:

(a) 9 with low complexity

In addition to above, system requires

i. Significant data communication

ii. Performance is very critical

iii. Designed code may be moderately reusable

iv. System is not designed for multiple installation in different organizations.

Other complexity adjustment factors are treated as average. Compute the function points for the project.

Function Point Analysis

Solution: Unadjusted function points may be counted using table 2

Functional Units	Count	Complexity		Complexity Totals	Functional Unit Totals
External Inputs (EIs)	10	Low x 3	=	30	
	15	Average x 4	=	60	
	17	High x 6	=	102	192
External Outputs (EOs)	6	Low x 4	=	24	
	0	Average x 5	=	0	
	13	High x 7	=	91	115
External Inquiries (EQs)	3	Low x 3	=	9	
	4	Average x 4	=	16	
	2	High x 6	=	12	37
External logical Files (ILFs)	0	Low x 7	=	0	
	2	Average x 10	=	20	
	1	High x 15	=	15	35
External Interface Files (EIFs)	9	Low x 5	=	45	
	0	Average x 7	=	0	
	0	High x 10	=	0	45
Total Unadjusted Function Point Count					424

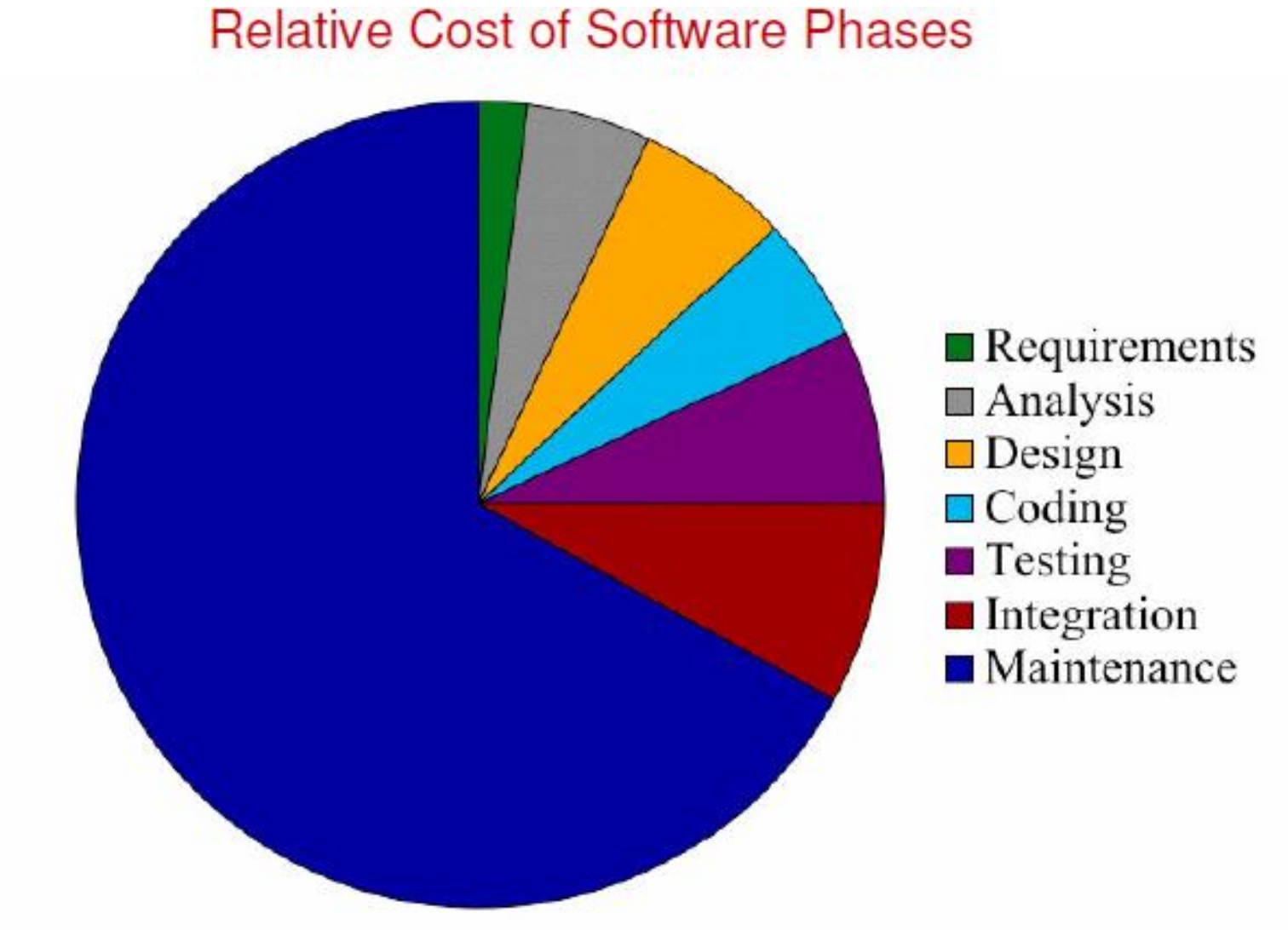
Function Point Analysis

$$\sum_{i=1}^{14} F_i = 3+4+3+5+3+3+3+3+3+3+2+3+0+3=41$$

$$\begin{aligned} CAF &= (0.65 + 0.01 \times \Sigma F_i) \\ &= (0.65 + 0.01 \times 41) \\ &= 1.06 \\ FP &= UFP \times CAF \\ &= 424 \times 1.06 \\ &= 449.44 \end{aligned}$$

Hence $\boxed{FP = 449}$

Software Project Planning



Heuristic Techniques

- Heuristic techniques assume that the relationships among the different project parameters can be modeled using suitable mathematical expressions. Once the basic (independent) parameters are known, the other (dependent) parameters can be easily determined by substituting the value of the basic parameters in the mathematical expression. Different heuristic estimation models can be divided into the following two classes: single variable model and the multi variable model.

Single variable model

Methods using this model use an equation to estimate the desired values such as cost, time, effort, etc. They all depend on the same variable used as predictor (say, size). An example of the most common equations is :

$$C = a\,L^{b}$$

SEL (Software Engineering Laboratory) model

C is the cost, L is the size and a, b are constants

$$E = 1.4\,L^{0.93}$$
$$DOC = 30.4\,L^{0.90}$$
$$D = 4.6\,L^{0.26}$$

Effort (E in Person-months), documentation (DOC, in number of pages) and duration (D, in months) are calculated from the number of lines of code (L, in thousands of lines) used as a predictor.

Single variable model

WF (Walston-Felix) model

These models are often based on equation, they actually depend on several variables representing various aspects of the software development environment, for example method used, user participation, customer oriented changes, memory constraints, etc.

$$E = 5.2\ L^{0.91}$$

$$D = 4.1\ L^{0.36}$$

Multi variable model

- The productivity index uses 29 variables which are found to be highly correlated to productivity as follows:

$$I = \sum_{i=1}^{29} W_i X_i$$

Heuristic Techniques

- Compare the WF (Walston-Felix) model with the SEL (Software Engineering Laboratory) model on a software development expected to involve 8 person-years of effort. Software Project Planning

(a) Calculate the number of lines of source code that can be produced.

(b) Calculate the duration of the development.

(c) Calculate the productivity in LOC/PY

(d) Calculate the average manning

Heuristic Techniques

Solution

The amount of manpower involved = 8 PY = 96 person-months

(a) Number of lines of source code can be obtained by reversing equation to give:

$$L = (E/a)^{1/b}$$

Then

$$L(SEL) = (96/1.4)^{1/0.93} = 94264 \text{ LOC}$$

$$L_{(W-F)} = (96/5.2)^{1/0.91} = 24632 \text{ LOC}.$$

Heuristic Techniques

(b) Duration in months can be calculated by means of equation

$$D(SEL) = 4.6\ (L)^{0.26}$$

$$= 4.6\ (94.264)^{0.26} = 15 \text{ months}$$

$$D(W\text{-}F) = 4.1\ L^{0.36}$$

$$= 4.1(24.632)^{0.36} = 13 \text{ months}$$

(c) Productivity is the lines of code produced per person/month (year)

$$P(SEL) = \frac{94264}{8} = 11783\ LOC / Person - Years$$

$$P(W-F) = \frac{24632}{8} = 3079\ LOC / Person - Years$$

Heuristic Techniques

(d) Average manning is the average number of persons required per month in the project.

$$M(SEL) = \frac{96P - M}{15M} = 6.4 Persons$$

$$M(W - F) = \frac{96P - M}{13M} = 7.4 Persons$$