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.

Software Project Planning

In order to conduct a successful software project, we must understand:

- Scope of work to be done
- 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.

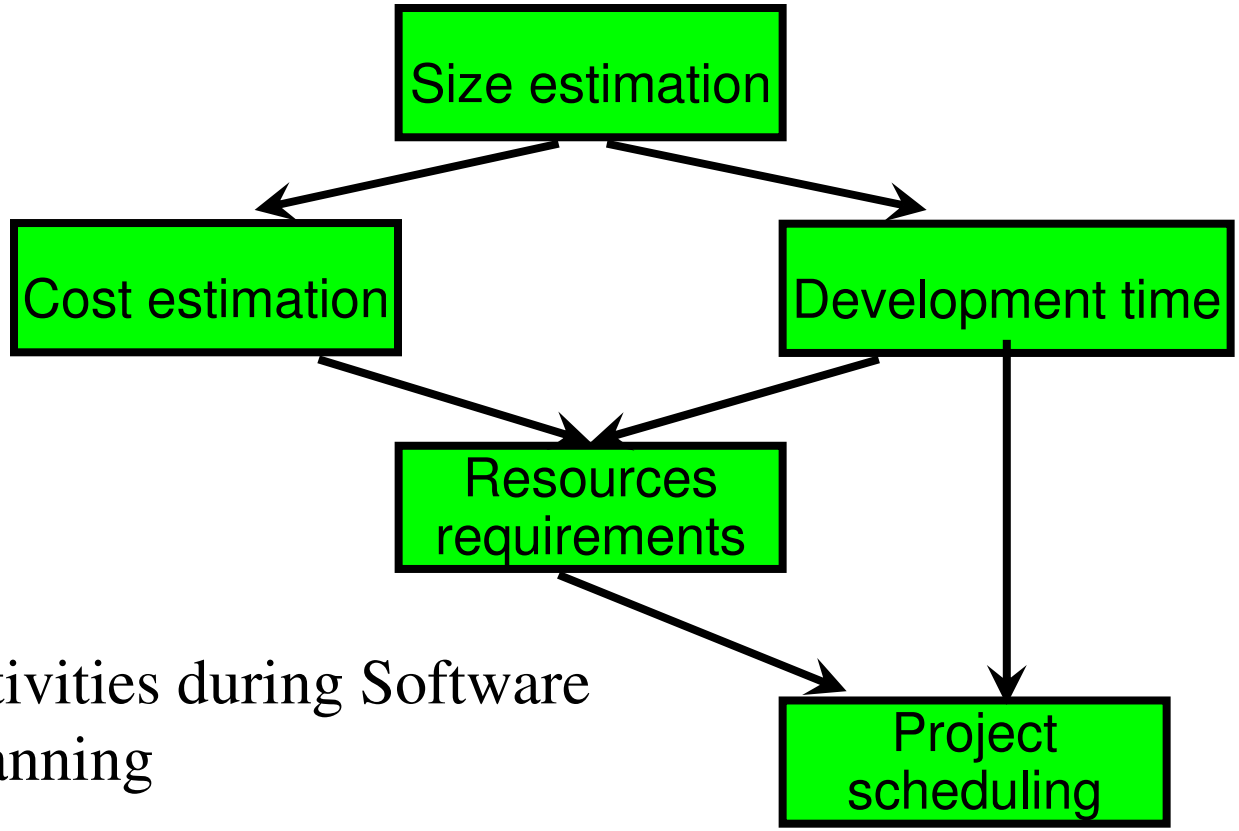


Fig. 1: Activities during Software Project Planning

Software Project Planning

Size Estimation

Lines of Code (LOC)

If LOC is simply a count of the number of lines then figure shown below contains 18 LOC .

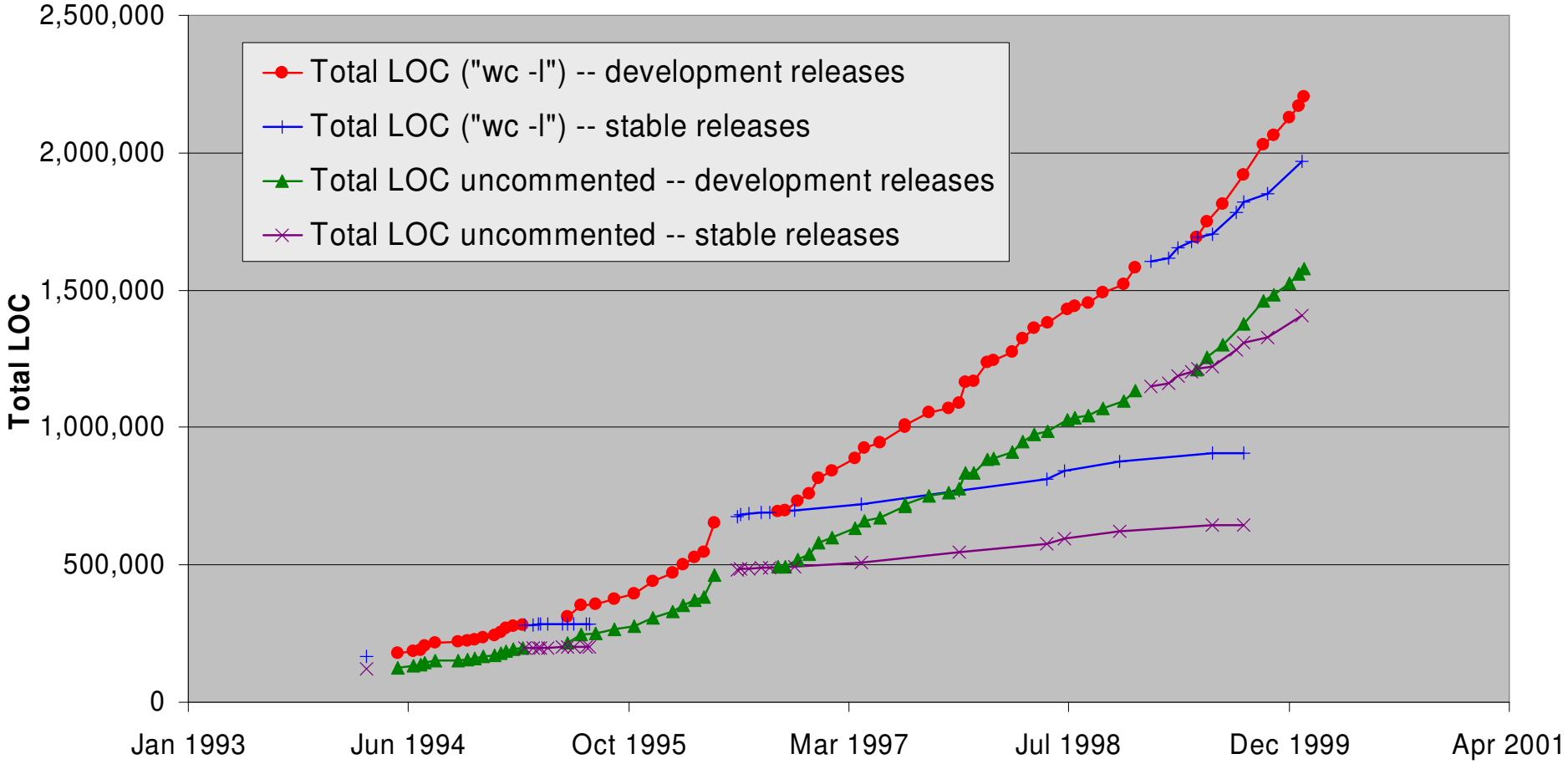
When comments and blank lines are ignored, the program in figure 2 shown below contains 17 LOC.

Fig. 2: Function for sorting an array

```
1.  int. sort (int x[ ], int n)
2.  {
3.  int i, j, save, im1;
4.  /*This function sorts array x in ascending order */
5.  If (n<2) return 1;
6.  for (i=2; i<=n; i++)
7.  {
8.  im1=i-1;
9.  for (j=1; j<=im; j++)
10. if (x[i] < x[j])
11. {
12. Save = x[i];
13. x[i] = x[j];
14. x[j] = save;
15. }
16. }
17. return 0;
18. }
```

Software Project Planning

Growth of Lines of Code (LOC)



Software Project Planning

Furthermore, if the main interest is the size of the program for specific functionality, it may be reasonable to include executable statements. The only executable statements in figure shown above are in lines 5-17 leading to a count of 13. The differences in the counts are 18 to 17 to 13. One can easily see the potential for major discrepancies for large programs with many comments or programs written in language that allow a large number of descriptive but non-executable statement. Conte has defined lines of code as:

Software Project Planning

"A line of code is any line of program text that is not a comment or blank line, regardless of the number of statements or fragments of statements on the line. This specifically includes all lines containing program header, declaration, and executable and non-executable statements".

This is the predominant definition for lines of code used by researchers. By this definition, figure shown above has 17 LOC.

Function Count

Alan Albrecht while working for IBM, recognized the problem in size measurement in the 1970s, and developed a technique (which he called Function Point Analysis), which appeared to be a solution to the size measurement problem.

Software Project Planning

The principle of Albrecht's function point analysis (FPA) is that a system is decomposed into functional units.

- Inputs : information entering the system
- Outputs : information leaving the system
- Enquiries : requests for instant access to information
- Internal logical files : information held within the system
- External interface files : information held by other system that is used by the system being analyzed.

Software Project Planning

The FPA functional units are shown in figure given below:

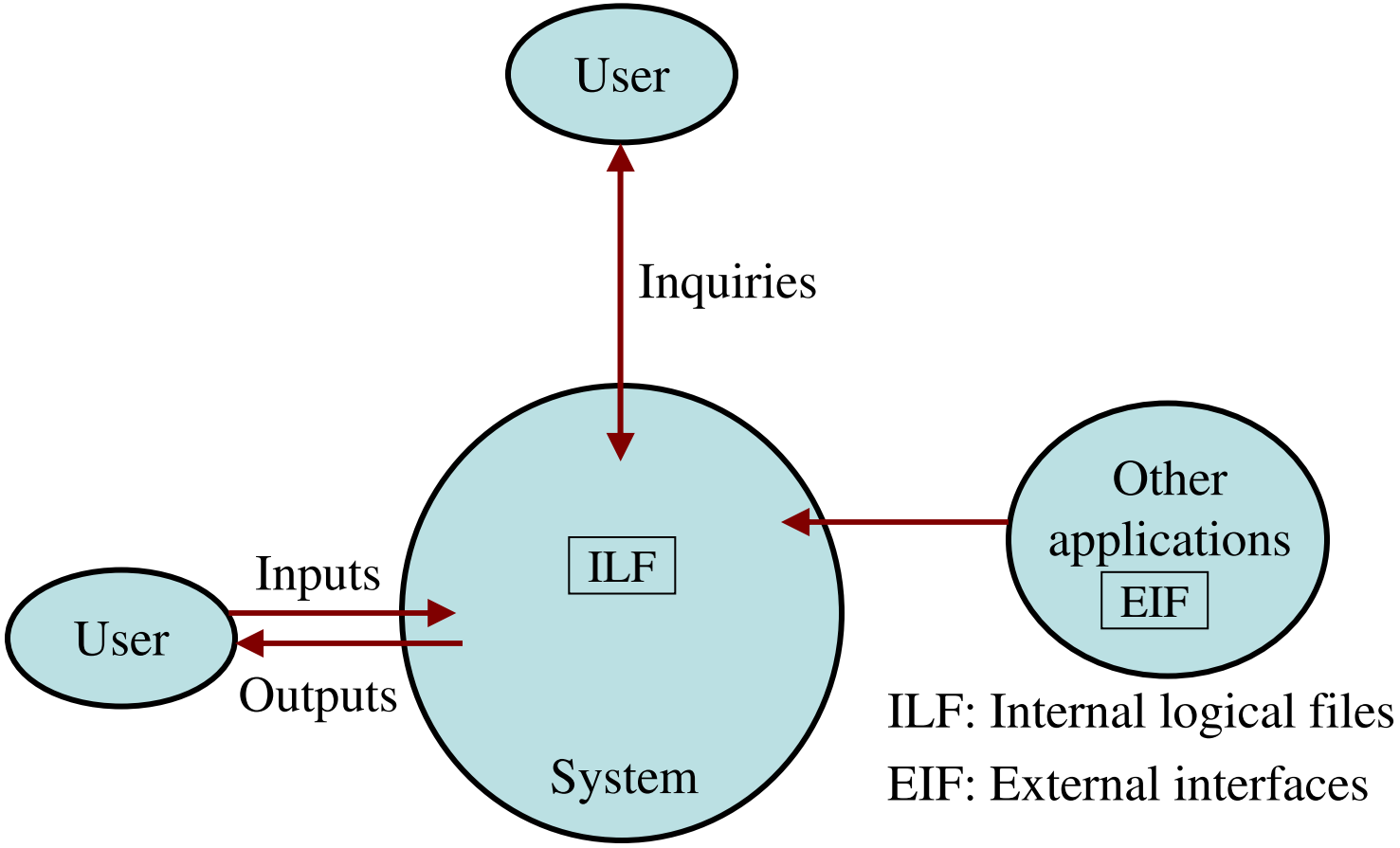


Fig. 3: FPAs functional units System

Software Project Planning

The five functional units are divided in two categories:

(i) Data function types

- Internal Logical Files (ILF): A user identifiable group of logical related data or control information maintained within the system.
- External Interface files (EIF): A user identifiable group of logically related data or control information referenced by the system, but maintained within another system. This means that EIF counted for one system, may be an ILF in another system.

Software Project Planning

(ii) Transactional function types

- External Input (EI): An EI processes data or control information that comes from outside the system. The EI is an elementary process, which is the smallest unit of activity that is meaningful to the end user in the business.
- External Output (EO): An EO is an elementary process that generate data or control information to be sent outside the system.
- External Inquiry (EQ): An EQ is an elementary process that is made up to an input-output combination that results in data retrieval.

Software Project Planning

Special features

- Function point approach is independent of the language, tools, or methodologies used for implementation; i.e. they do not take into consideration programming languages, data base management systems, processing hardware or any other data base technology.
- Function points can be estimated from requirement specification or design specification, thus making it possible to estimate development efforts in early phases of development.

Software Project Planning

- Function points are directly linked to the statement of requirements; any change of requirements can easily be followed by a re-estimate.
- Function points are based on the system user's external view of the system, non-technical users of the software system have a better understanding of what function points are measuring.

Software Project Planning

Counting function points

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

Software Project Planning

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				

Software Project Planning

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.

Software Project Planning

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

Zij : 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*.

Software Project Planning

Organizations that use function point methods develop a criterion for determining whether a particular entry is Low, Average or High. Nonetheless, the determination of complexity is somewhat subjective.

$$FP = UFP * CAF$$

Where CAF is complexity adjustment factor and is equal to [0.65 + 0.01 x ΣF_i]. The F_i (i=1 to 14) are the degree of influence and are based on responses to questions noted in table 3.

Table 3 : Computing function points.

Rate each factor on a scale of 0 to 5.

Number of factors considered (F_i)

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 ?
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 ?

Software Project Planning

Functions points may compute the following important metrics:

Productivity	=	FP / persons-months
Quality	=	Defects / FP
Cost	=	Rupees / FP
Documentation	=	Pages of documentation per FP

These metrics are controversial and are not universally acceptable. There are standards issued by the International Functions Point User Group (IFPUG, covering the Albrecht method) and the United Kingdom Function Point User Group (UFPGU, covering the MK11 method). An ISO standard for function point method is also being developed.

Software Project Planning

Example: 4.1

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.

Software Project Planning

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

Example:4.2

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 ?

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.

Software Project Planning

Example: 4.3

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

Software Project Planning

(*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.

Software Project Planning

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

Software Project Planning

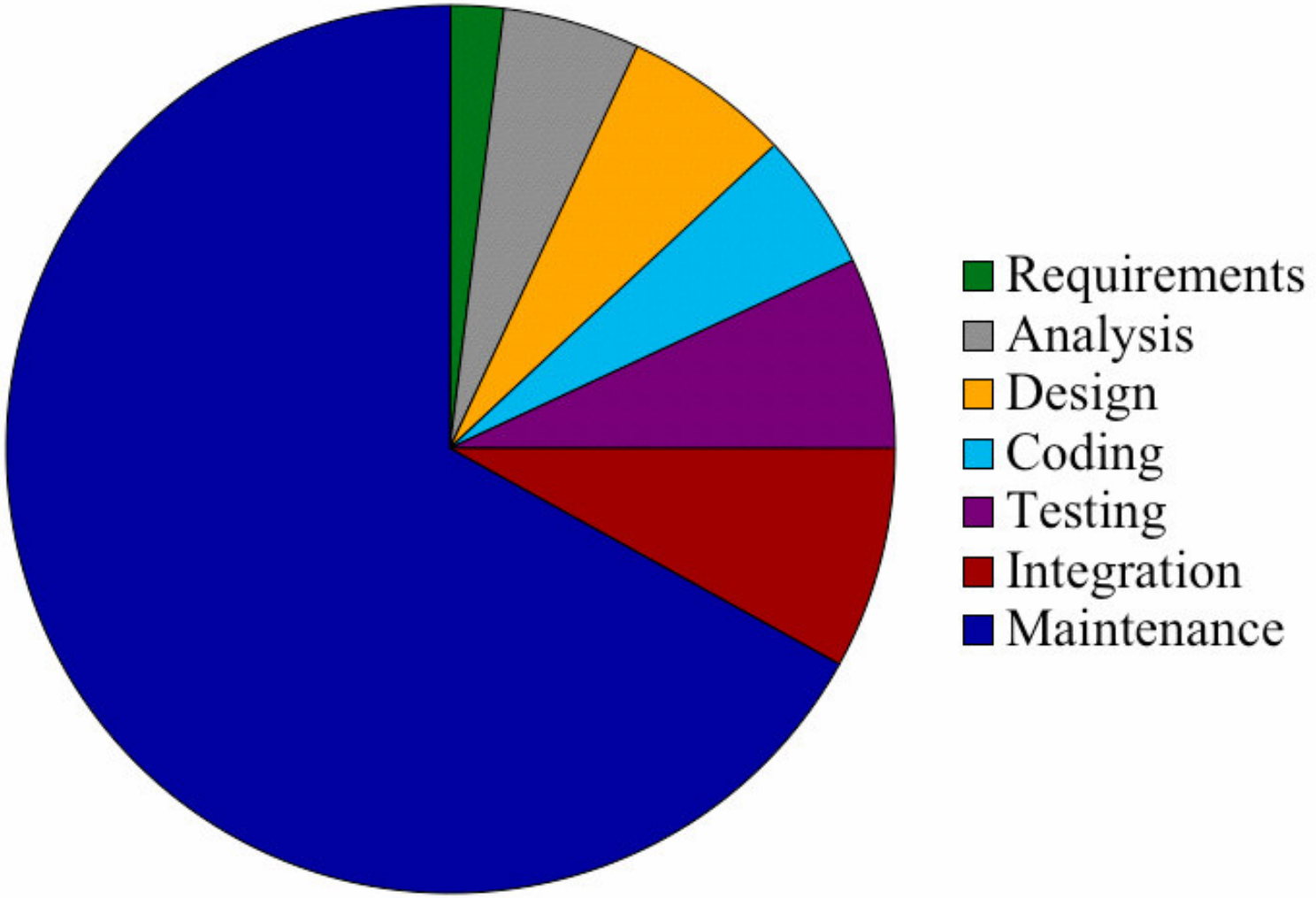
$$\sum_{i=1}^{14} F_i = 3+4+3+5+3+3+3+3+3+3+2+3+0+3=41$$

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

Hence $\boxed{FP = 449}$

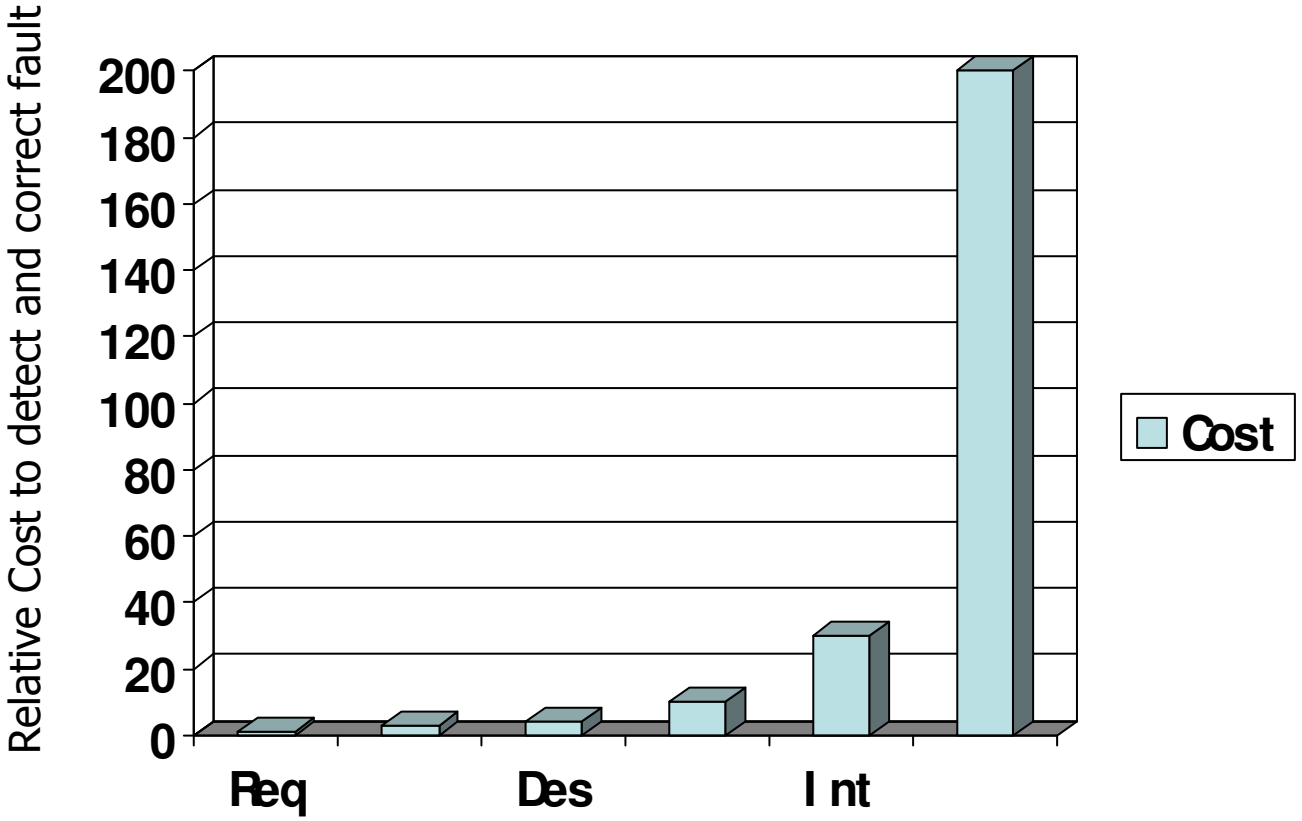
Software Project Planning

Relative Cost of Software Phases



Software Project Planning

Cost to Detect and Fix Faults



Software Project Planning

Cost Estimation

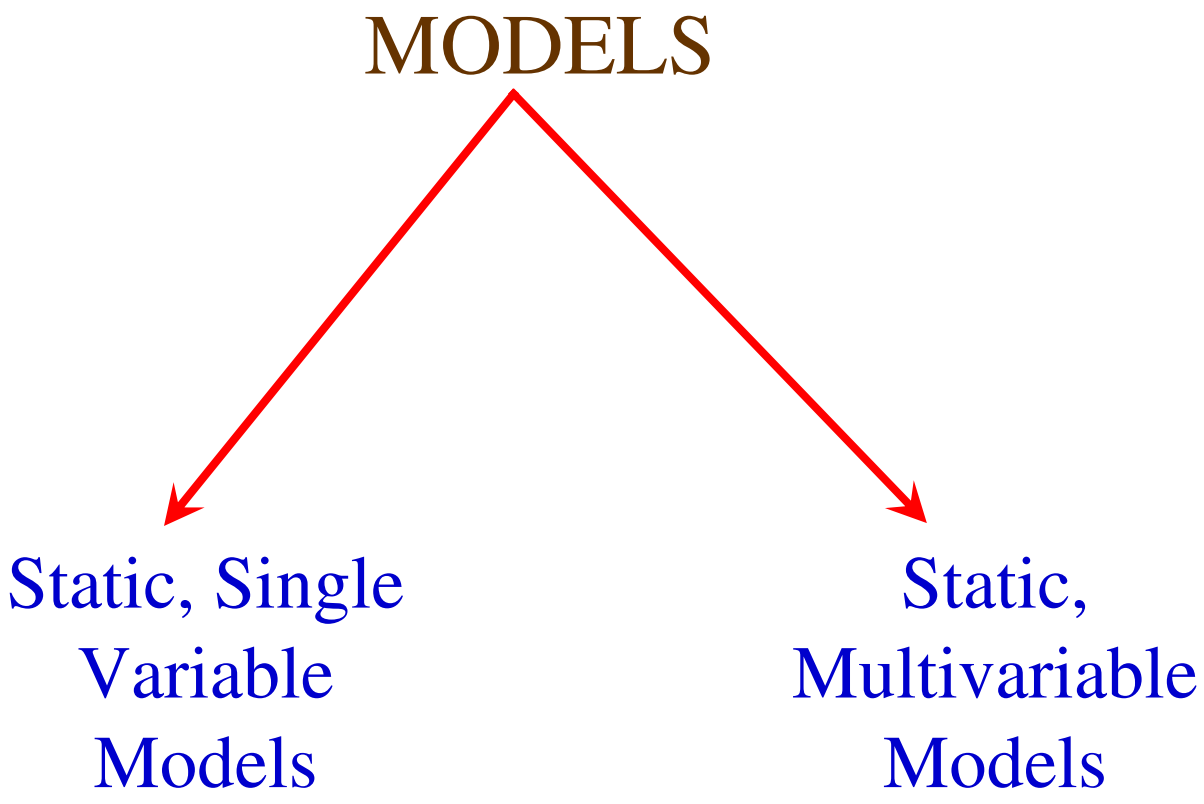
A number of estimation techniques have been developed and are having following attributes in common :

- Project scope must be established in advance
- Software metrics are used as a basis from which estimates are made
- The project is broken into small pieces which are estimated individually

To achieve reliable cost and schedule estimates, a number of options arise:

- Delay estimation until late in project
- Use simple decomposition techniques to generate project cost and schedule estimates
- Develop empirical models for estimation
- Acquire one or more automated estimation tools

Software Project Planning



Software Project Planning

Static, Single Variable Models

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 :

$$C = a L^b \quad \text{(i)}$$

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.

Software Project Planning

Static, Multivariable Models

These models are often based on equation (i), 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.

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

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

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

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

Software Project Planning

Example: 4.4

Compare the Walston-Felix model with the SEL model on a software development expected to involve 8 person-years of effort.

(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

Software Project Planning

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

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

$$L(SEL) = (96/5.2)^{1/0.91} = 24632 \text{ LOC}.$$

Software Project Planning

(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)

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

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

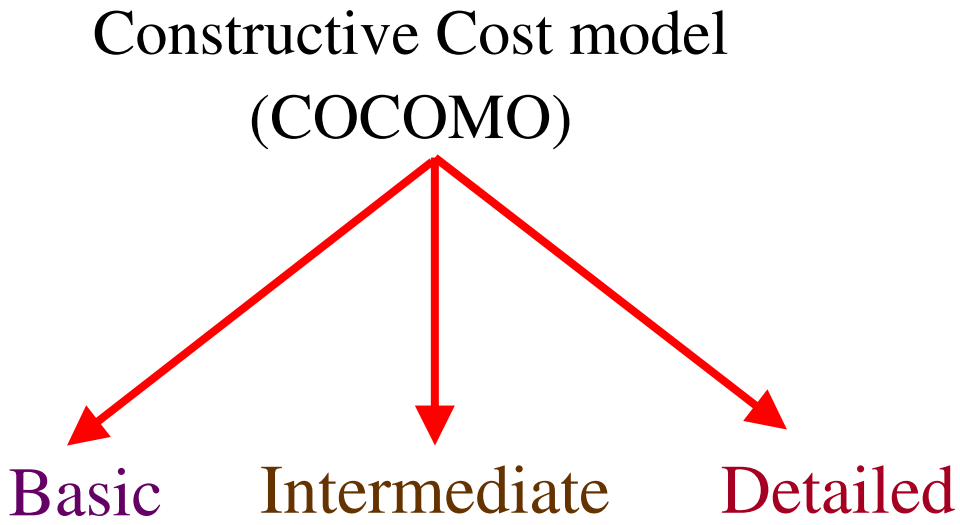
Software Project Planning

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

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

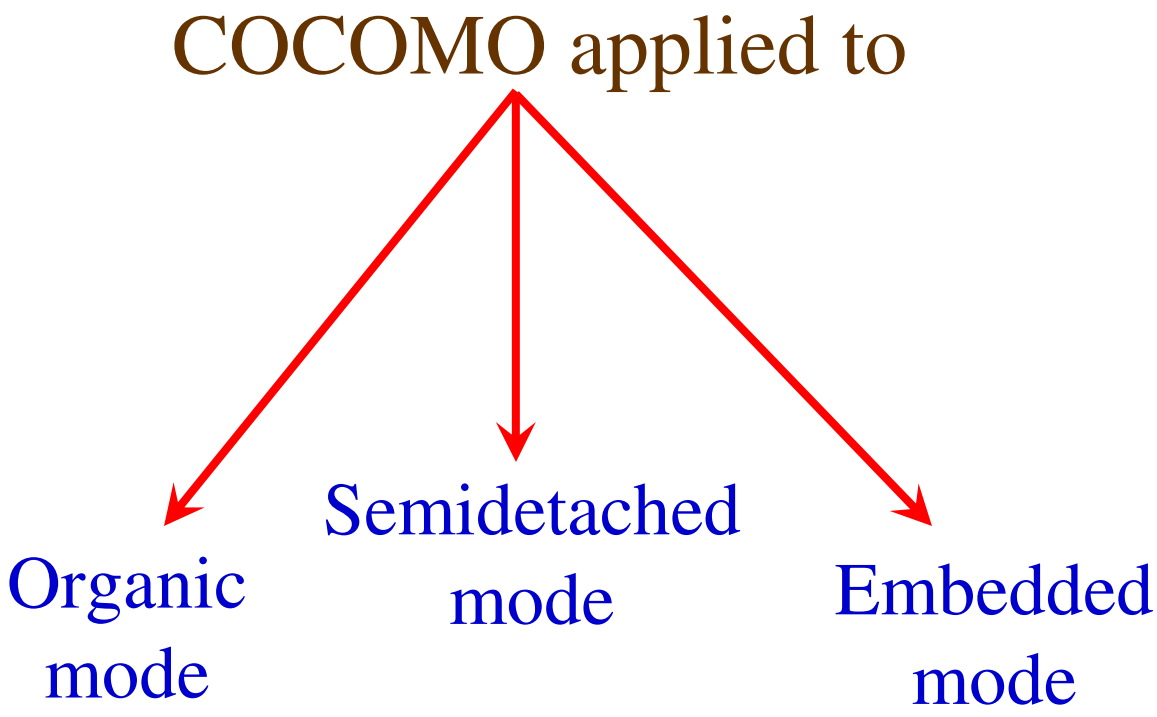
$$M(W - F) = \frac{96P - M}{13M} = 7.4 Persons$$

The Constructive Cost Model (COCOMO)



Model proposed by
B. W. Boehm's
through his book
Software Engineering Economics in 1981

Software Project Planning



Software Project Planning

Mode	Project size	Nature of Project	Innovation	Deadline of the project	Development Environment
Organic	Typically 2-50 KLOC	Small size project, experienced developers in the familiar environment. For example, pay roll, inventory projects etc.	Little	Not tight	Familiar & In house
Semi detached	Typically 50-300 KLOC	Medium size project, Medium size team, Average previous experience on similar project. For example: Utility systems like compilers, database systems, editors etc.	Medium	Medium	Medium
Embedded	Typically over 300 KLOC	Large project, Real time systems, Complex interfaces, Very little previous experience. For example: ATMs, Air Traffic Control etc.	Significant	Tight	Complex Hardware/ customer Interfaces required

Table 4: The comparison of three COCOMO modes

Software Project Planning

Basic Model

Basic COCOMO model takes the form

$$E = a_b (KLOC)^{b_b}$$

$$D = c_b (E)^{d_b}$$

where E is effort applied in Person-Months, and D is the development time in months. The coefficients a_b, b_b, c_b and d_b are given in table 4 (a).

Software Project Planning

Software Project	a_b	b_b	c_b	d_b
Organic	2.4	1.05	2.5	0.38
Semidetached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

Table 4(a): Basic COCOMO coefficients

Software Project Planning

When effort and development time are known, the average staff size to complete the project may be calculated as:

$$\text{Average staff size } (SS) = \frac{E}{D} \; Persons$$

When project size is known, the productivity level may be calculated as:

$$\text{Productivity } (P) = \frac{KLOC}{E} \; KLOC / PM$$

Software Project Planning

Example: 4.5

Suppose that a project was estimated to be 400 KLOC. Calculate the effort and development time for each of the three modes i.e., organic, semidetached and embedded.

Software Project Planning

Solution

The basic COCOMO equation take the form:

$$E = a_b (KLOC)^{b_b}$$

$$D = c_b (KLOC)^{d_b}$$

Estimated size of the project = 400 KLOC

(i) Organic mode

$$E = 2.4(400)^{1.05} = 1295.31 \text{ PM}$$

$$D = 2.5(1295.31)^{0.38} = 38.07 \text{ PM}$$

Software Project Planning

(ii) Semidetached mode

$$E = 3.0(400)^{1.12} = 2462.79 \text{ PM}$$

$$D = 2.5(2462.79)^{0.35} = 38.45 \text{ PM}$$

(iii) Embedded mode

$$E = 3.6(400)^{1.20} = 4772.81 \text{ PM}$$

$$D = 2.5(4772.8)^{0.32} = 38 \text{ PM}$$

Software Project Planning

Example: 4.6

A project size of 200 KLOC is to be developed. Software development team has average experience on similar type of projects. The project schedule is not very tight. Calculate the effort, development time, average staff size and productivity of the project.

Software Project Planning

Solution

The semi-detached mode is the most appropriate mode; keeping in view the size, schedule and experience of the development team.

Hence $E = 3.0(200)^{1.12} = 1133.12$ PM

$D = 2.5(1133.12)^{0.35} = 29.3$ PM

$$\text{Average staff size } (SS) = \frac{E}{D} Persons$$

$$= \frac{1133.12}{29.3} = 38.67 Persons$$

Software Project Planning

$$\text{Productivity} = \frac{KLOC}{E} = \frac{200}{1133.12} = 0.1765\ KLOC / PM$$

$$P = 176\ LOC / PM$$

Software Project Planning

Intermediate Model

Cost drivers

(*i*) Product Attributes

- Required s/w reliability
- Size of application database
- Complexity of the product

(*ii*) Hardware Attributes

- Run time performance constraints
- Memory constraints
- Virtual machine volatility
- Turnaround time

Software Project Planning

(*iii*) Personal Attributes

- Analyst capability
- Programmer capability
- Application experience
- Virtual m/c experience
- Programming language experience

(*iv*) Project Attributes

- Modern programming practices
- Use of software tools
- Required development Schedule

Software Project Planning

Multipliers of different cost drivers

Cost Drivers	RATINGS					
	Very low	Low	Nominal	High	Very high	Extra high
Product Attributes						
RELY	0.75	0.88	1.00	1.15	1.40	--
DATA	--	0.94	1.00	1.08	1.16	--
CPLX	0.70	0.85	1.00	1.15	1.30	1.65
Computer Attributes						
TIME	--	--	1.00	1.11	1.30	1.66
STOR	--	--	1.00	1.06	1.21	1.56
VIRT	--	0.87	1.00	1.15	1.30	--
TURN	--	0.87	1.00	1.07	1.15	--

Software Project Planning

Cost Drivers	RATINGS					
	Very low	Low	Nominal	High	Very high	Extra high
Personnel Attributes						
ACAP	1.46	1.19	1.00	0.86	0.71	--
AEXP	1.29	1.13	1.00	0.91	0.82	--
PCAP	1.42	1.17	1.00	0.86	0.70	--
VEXP	1.21	1.10	1.00	0.90	--	--
LEXP	1.14	1.07	1.00	0.95	--	--
Project Attributes						
MODP	1.24	1.10	1.00	0.91	0.82	--
TOOL	1.24	1.10	1.00	0.91	0.83	--
SCED	1.23	1.08	1.00	1.04	1.10	--

Table 5: Multiplier values for effort calculations

Software Project Planning

Intermediate COCOMO equations

$$E = a_i (KLOC)^{b_i} * EAF$$

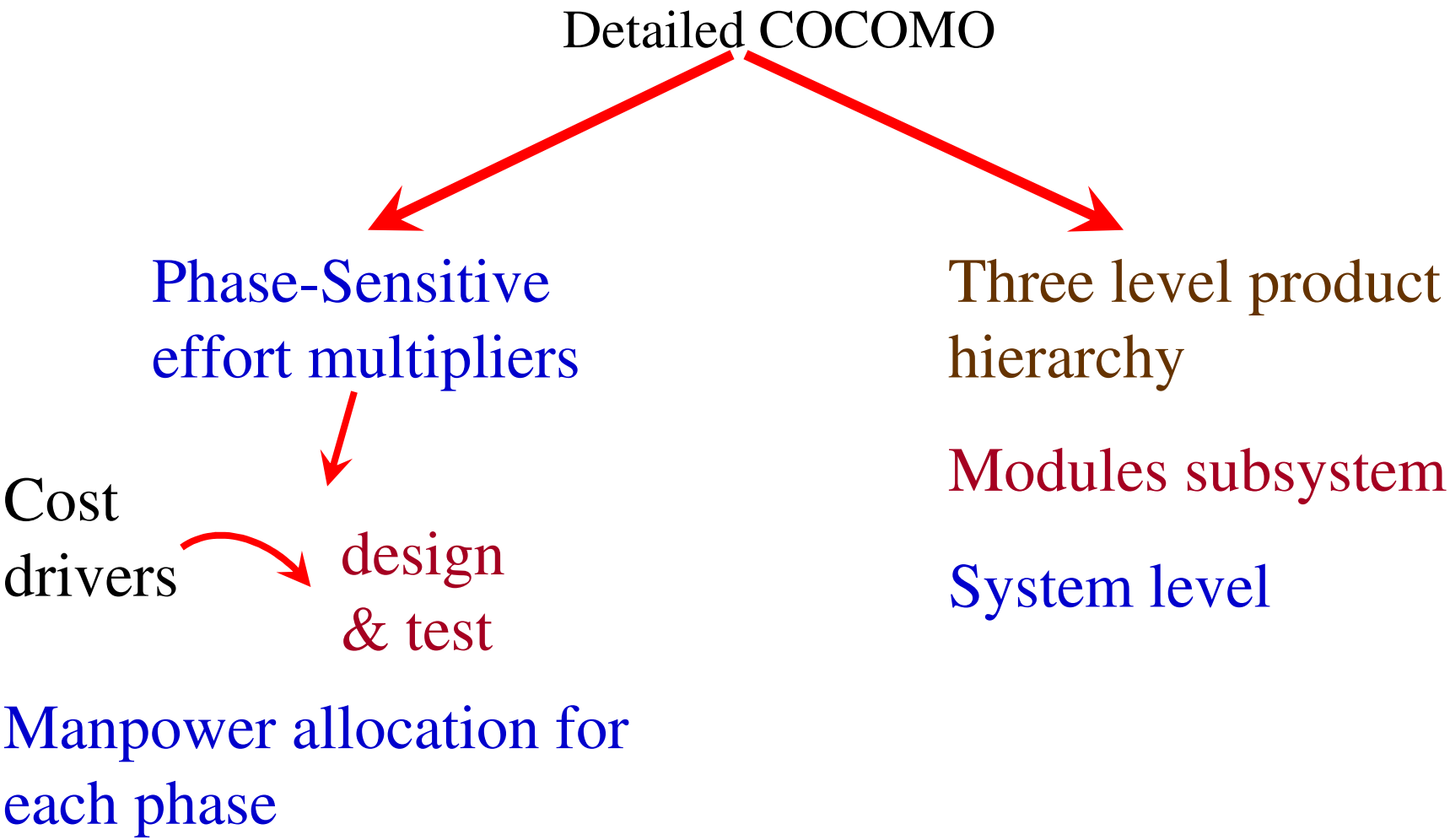
$$D = c_i (E)^{d_i}$$

Project	a_i	b_i	c_i	d_i
Organic	3.2	1.05	2.5	0.38
Semidetached	3.0	1.12	2.5	0.35
Embedded	2.8	1.20	2.5	0.32

Table 6: Coefficients for intermediate COCOMO

Software Project Planning

Detailed COCOMO Model



Software Project Planning

Development Phase

Plan / Requirements

EFFORT	:	6% to 8%
DEVELOPMENT TIME	:	10% to 40%

% depend on mode & size

Software Project Planning

Design

Effort : 16% to 18%

Time : 19% to 38%

Programming

Effort : 48% to 68%

Time : 24% to 64%

Integration & Test

Effort : 16% to 34%

Time : 18% to 34%

Software Project Planning

Principle of the effort estimate

Size equivalent

As the software might be partly developed from software already existing (that is, re-usable code), a full development is not always required. In such cases, the parts of design document (DD%), code (C%) and integration (I%) to be modified are estimated. Then, an adjustment factor, A, is calculated by means of the following equation.

$$A = 0.4\ DD + 0.3\ C + 0.3\ I$$

The size equivalent is obtained by

$$S\ (equivalent) = (S \times A) / 100$$

$$E_p = \mu_p E$$

$$D_p = \tau_p D$$

Software Project Planning

Lifecycle Phase Values of μ_p

Mode & Code Size	Plan & Requirements	System Design	Detailed Design	Module Code & Test	Integration & Test
Organic Small S≈2	0.06	0.16	0.26	0.42	0.16
Organic medium S≈32	0.06	0.16	0.24	0.38	0.22
Semidetached medium S≈32	0.07	0.17	0.25	0.33	0.25
Semidetached large S≈128	0.07	0.17	0.24	0.31	0.28
Embedded large S≈128	0.08	0.18	0.25	0.26	0.31
Embedded extra large S≈320	0.08	0.18	0.24	0.24	0.34

Table 7 : Effort and schedule fractions occurring in each phase of the lifecycle

Software Project Planning

Lifecycle Phase Values of τ_p

Mode & Code Size	Plan & Requirements	System Design	Detailed Design	Module Code & Test	Integration & Test
Organic Small S≈2	0.10	0.19	0.24	0.39	0.18
Organic medium S≈32	0.12	0.19	0.21	0.34	0.26
Semidetached medium S≈32	0.20	0.26	0.21	0.27	0.26
Semidetached large S≈128	0.22	0.27	0.19	0.25	0.29
Embedded large S≈128	0.36	0.36	0.18	0.18	0.28
Embedded extra large S≈320	0.40	0.38	0.16	0.16	0.30

Table 7 : Effort and schedule fractions occurring in each phase of the lifecycle

Software Project Planning

Distribution of software life cycle:

1. Requirement and product design
   (a) Plans and requirements
   (b) System design
2. Detailed Design
   (a) Detailed design
3. Code & Unit test
   (a) Module code & test
4. Integrate and Test
   (a) Integrate & Test

Software Project Planning

Example: 4.7

A new project with estimated 400 KLOC embedded system has to be developed. Project manager has a choice of hiring from two pools of developers: Very highly capable with very little experience in the programming language being used

Or

Developers of low quality but a lot of experience with the programming language. What is the impact of hiring all developers from one or the other pool ?

Software Project Planning

Solution

This is the case of embedded mode and model is intermediate COCOMO.

Hence $$E = a_i (KLOC)^{d_i}$$

$$= 2.8\ (400)^{1.20} = 3712\ PM$$

Case I: Developers are very highly capable with very little experience in the programming being used.

EAF = 0.82 x 1.14 = 0.9348

E = 3712 x .9348 = 3470 PM

D = $2.5\ (3470)^{0.32}$ = 33.9 M

Software Project Planning

Case II: Developers are of low quality but lot of experience with the programming language being used.

EAF = 1.29 x 0.95 = 1.22

E = 3712 x 1.22 = 4528 PM

D = $2.5 (4528)^{0.32}$ = 36.9 M

Case II requires more effort and time. Hence, low quality developers with lot of programming language experience could not match with the performance of very highly capable developers with very litter experience.

Software Project Planning

Example: 4.8

Consider a project to develop a full screen editor. The major components identified are:

I. Screen edit

II. Command Language Interpreter

III. File Input & Output

IV. Cursor Movement

V. Screen Movement

The size of these are estimated to be 4k, 2k, 1k, 2k and 3k delivered source code lines. Use COCOMO to determine

1. Overall cost and schedule estimates (assume values for different cost drivers, with at least three of them being different from 1.0)
2. Cost & Schedule estimates for different phases.

Software Project Planning

Solution

Size of five modules are:

Screen edit	= 4 KLOC
Command language interpreter	= 2 KLOC
File input and output	= 1 KLOC
Cursor movement	= 2 KLOC
Screen movement	= 3 KLOC
Total	**= 12 KLOC**

Software Project Planning

Let us assume that significant cost drivers are

i. Required software reliability is high, i.e.,1.15

ii. Product complexity is high, i.e.,1.15

iii. Analyst capability is high, i.e.,0.86

iv. Programming language experience is low,i.e.,1.07

v. All other drivers are nominal

EAF = 1.15x1.15x0.86x1.07 = 1.2169

Software Project Planning

(a) The initial effort estimate for the project is obtained from the following equation

$$E = a_i (KLOC)^{bi} \times EAF$$

$$= 3.2(12)^{1.05} \times 1.2169 = 52.91 \text{ PM}$$

Development time $D = C_i(E)^{di}$

$$= 2.5(52.91)^{0.38} = 11.29 \text{ M}$$

(b) Using the following equations and referring Table 7, phase wise cost and schedule estimates can be calculated.

$$E_p = \mu_p E$$

$$D_p = \tau_p D$$

Software Project Planning

Since size is only 12 KLOC, it is an organic small model. Phase wise effort distribution is given below:

System Design	= 0.16 x 52.91 = 8.465 PM
Detailed Design	= 0.26 x 52.91 = 13.756 PM
Module Code & Test	= 0.42 x 52.91 = 22.222 PM
Integration & Test	= 0.16 x 52.91 = 8.465 Pm

Now Phase wise development time duration is

System Design	= 0.19 x 11.29 = 2.145 M
Detailed Design	= 0.24 x 11.29 = 2.709 M
Module Code & Test	= 0.39 x 11.29 = 4.403 M
Integration & Test	= 0.18 x 11.29 = 2.032 M

Software Project Planning

COCOMO-II

The following categories of applications / projects are identified by COCOMO-II and are shown in fig. 4 shown below:

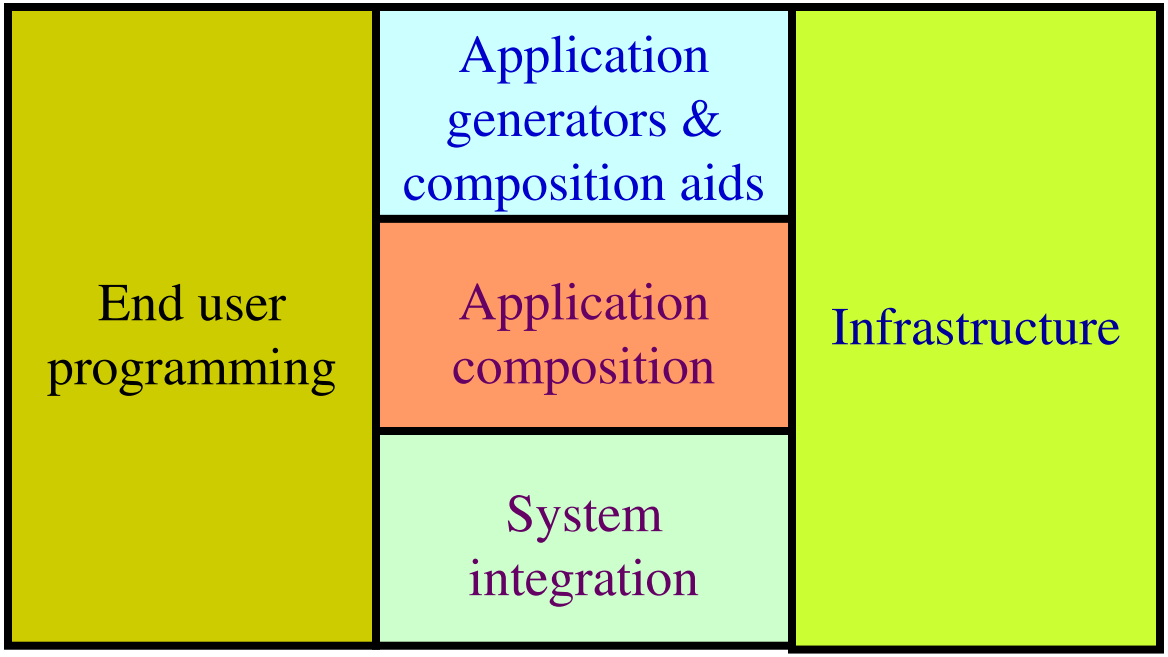


Fig. 4 : Categories of applications / projects

Software Project Planning

Stage No	Model Name	Application for the types of projects	Applications
Stage I	Application composition estimation model	Application composition	In addition to application composition type of projects, this model is also used for prototyping (if any) stage of application generators, infrastructure & system integration.
Stage II	Early design estimation model	Application generators, infrastructure & system integration	Used in early design stage of a project, when less is known about the project.
Stage III	Post architecture estimation model	Application generators, infrastructure & system integration	Used after the completion of the detailed architecture of the project.

Table 8: Stages of COCOMO-II

Application Composition Estimation Model

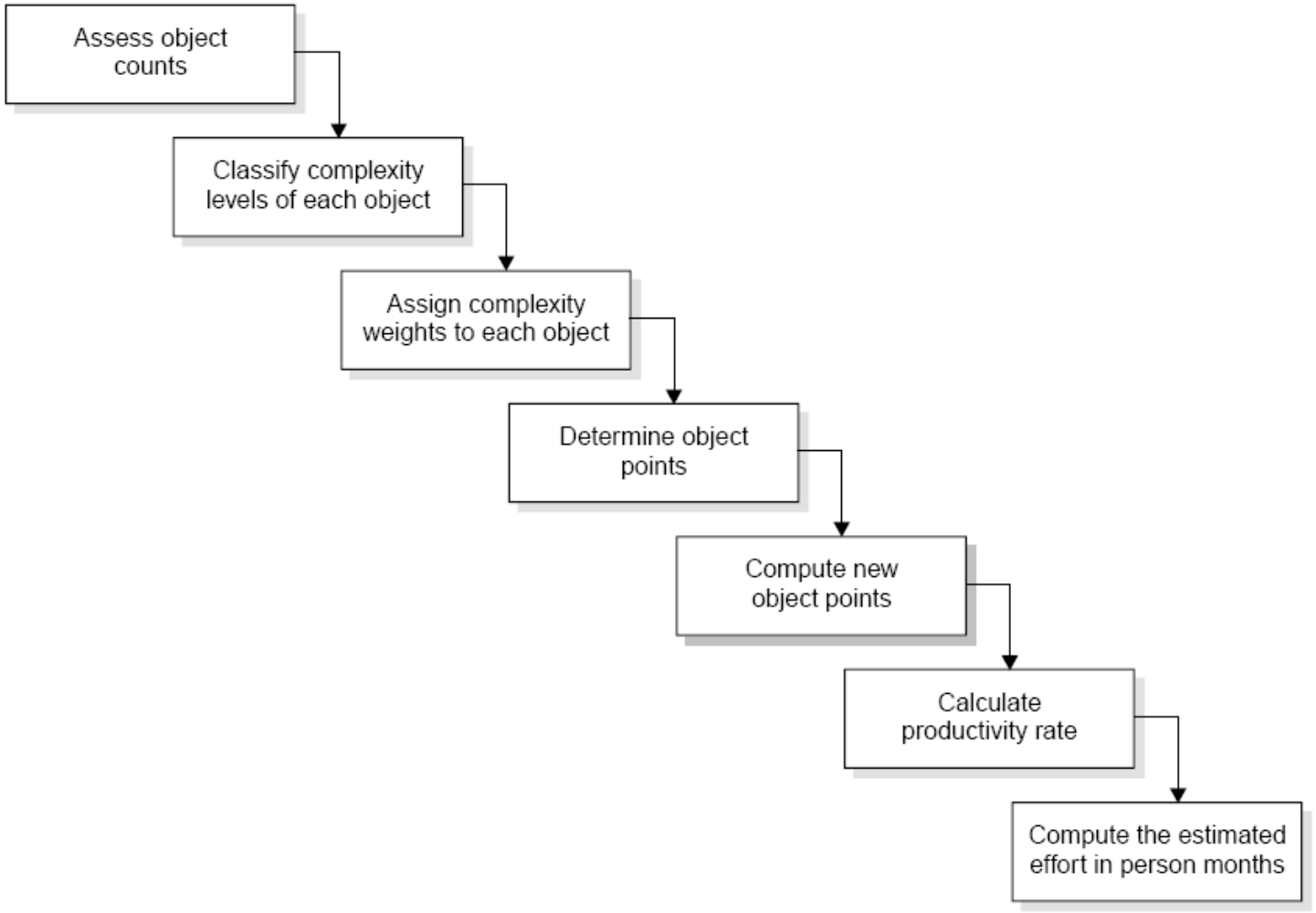


Fig.5: Steps for the estimation of effort in person months

Software Project Planning

i. Assess object counts: Estimate the number of screens, reports and 3 GL components that will comprise this application.

ii. Classification of complexity levels: We have to classify each object instance into simple, medium and difficult complexity levels depending on values of its characteristics.

Number of views contained	*# and sources of data tables*		
	Total < 4 (< 2 server < 3 client)	*Total < 8 (2 – 3 server 3 – 5 client)*	*Total 8 + (> 3 server, > 5 client)*
< 3	Simple	Simple	Medium
3 – 7	Simple	Medium	Difficult
> 8	Medium	Difficult	Difficult

Table 9 (a): For screens

Software Project Planning

Number of sections contained	*# and sources of data tables*		
	Total < 4 (< 2 server < 3 client)	*Total < 8 (2 – 3 server 3 – 5 client)*	*Total 8 + (> 3 server, > 5 client)*
0 or 1	Simple	Simple	Medium
2 or 3	Simple	Medium	Difficult
4 +	Medium	Difficult	Difficult

Table 9 (b): For reports

Software Project Planning

iii. Assign complexity weight to each object : The weights are used for three object types i.e., screen, report and 3GL components using the Table 10.

Object Type	*Complexity Weight*		
	Simple	*Medium*	*Difficult*
Screen	1	2	3
Report	2	5	8
3GL Component	—	—	10

Table 10: Complexity weights for each level

Software Project Planning

iv. Determine object points: Add all the weighted object instances to get one number and this known as object-point count.

v. Compute new object points: We have to estimate the percentage of reuse to be achieved in a project. Depending on the percentage reuse, the new object points (NOP) are computed.

$$\textbf{NOP} = \frac{(\text{object points}) * (100 - \%\text{reuse})}{100}$$

NOP are the object points that will need to be developed and differ from the object point count because there may be reuse.

Software Project Planning

vi. Calculation of productivity rate: The productivity rate can be calculated as:

Productivity rate (PROD) = NOP/Person month

Developer's experience & capability; ICASE maturity & capability	*PROD (NOP/PM)*
Very low	4
Low	7
Nominal	13
High	25
Very high	50

Table 11: Productivity values

Software Project Planning

vii. Compute the effort in Persons-Months: When PROD is known, we may estimate effort in Person-Months as:

$$\textbf{Effort in PM} = \frac{\text{NOP}}{\text{PROD}}$$

Software Project Planning

Example: 4.9

Consider a database application project with the following characteristics:

I. The application has 4 screens with 4 views each and 7 data tables for 3 servers and 4 clients.

II. The application may generate two report of 6 sections each from 07 data tables for two server and 3 clients. There is 10% reuse of object points.

The developer's experience and capability in the similar environment is low. The maturity of organization in terms of capability is also low. Calculate the object point count, New object points and effort to develop such a project.

Software Project Planning

Solution

This project comes under the category of application composition estimation model.

Number of screens = 4 with 4 views each

Number of reports = 2 with 6 sections each

From Table 9 we know that each screen will be of medium complexity and each report will be difficult complexity.

Using Table 10 of complexity weights, we may calculate object point count.

$$= 4 \times 2 + 2 \times 8 = 24$$

$$\mathbf{NOP} = \frac{24 * (100 - 10)}{100} = 21.6$$

Software Project Planning

Table 11 gives the low value of productivity (PROD) i.e. 7.

$$\text{Efforts in PM} = \frac{\text{NOP}}{\text{PROD}}$$

$$\text{Efforts} = \frac{21.6}{7} = 3.086 \text{ PM}$$

Software Project Planning

The Early Design Model

The COCOMO-II models use the base equation of the form

$$PM_{nominal} = A * (size)^B$$

where

$PM_{nominal}$ = Effort of the project in person months

A = Constant representing the nominal productivity, provisionally set to 2.5

B = Scale factor

Size = Software size

Software Project Planning

Scale factor	Explanation	Remarks
Precedentness	Reflects the previous experience on similar projects. This is applicable to individuals & organization both in terms of expertise & experience	Very low means no previous experiences, Extra high means that organization is completely familiar with this application domain.
Development flexibility	Reflect the degree of flexibility in the development process.	Very low means a well defined process is used. Extra high means that the client gives only general goals.
Architecture/ Risk resolution	Reflect the degree of risk analysis carried out.	Very low means very little analysis and Extra high means complete and through risk analysis.

Cont…

Table 12: Scaling factors required for the calculation of the value of B

Software Project Planning

Scale factor	Explanation	Remarks
Team cohesion	Reflects the team management skills.	Very low means no previous experiences, Extra high means that organization is completely familiar with this application domain.
Process maturity	Reflects the process maturity of the organization. Thus it is dependent on SEI-CMM level of the organization.	Very low means organization has no level at all and extra high means organization is related as highest level of SEI-CMM.

Table 12: Scaling factors required for the calculation of the value of B

Software Project Planning

Scaling factors	Very low	Low	Nominal	High	Very high	Extra high
Precedent ness	6.20	4.96	3.72	2.48	1.24	0.00
Development flexibility	5.07	4.05	3.04	2.03	1.01	0.00
Architecture/ Risk resolution	7.07	5.65	4.24	2.83	1.41	0.00
Team cohesion	5.48	4.38	3.29	2.19	1.10	0.00
Process maturity	7.80	6.24	4.68	3.12	1.56	0.00

Table 13: Data for the Computation of B

The value of B can be calculated as:

$$B=0.91 + 0.01 * (\text{Sum of rating on scaling factors for the project})$$

Software Project Planning

Early design cost drivers

There are seven early design cost drivers and are given below:

i. Product Reliability and Complexity (RCPX)

ii. Required Reuse (RUSE)

iii. Platform Difficulty (PDIF)

iv. Personnel Capability (PERS)

v. Personnel Experience (PREX)

vi. Facilities (FCIL)

vii. Schedule (SCED)

Software Project Planning

Post architecture cost drivers

There are 17 cost drivers in the Post Architecture model. These are rated on a scale of 1 to 6 as given below :

Very Low	*Low*	*Nominal*	*High*	*Very High*	*Extra High*
1	2	3	4	5	6

The list of seventeen cost drivers is given below :

i. Reliability Required (RELY)

ii. Database Size (DATA)

iii. Product Complexity (CPLX)

iv. Required Reusability (RUSE)

Software Project Planning

v. Documentation (DOCU)

vi. Execution Time Constraint (TIME)

vii. Main Storage Constraint (STOR)

viii. Platform Volatility (PVOL)

ix. Analyst Capability (ACAP)

x. Programmers Capability (PCAP)

xi. Personnel Continuity (PCON)

xii. Analyst Experience (AEXP)

Software Project Planning

xiii. Programmer Experience (PEXP)

xiv. Language & Tool Experience (LTEX)

xv. Use of Software Tools (TOOL)

xvi. Site Locations & Communication Technology between Sites (SITE)

xvii. Schedule (SCED)

Mapping of early design cost drivers and post architecture cost drivers

The 17 Post Architecture Cost Drivers are mapped to 7 Early Design Cost Drivers and are given in Table 14

Early Design Cost Drivers	Counter part Combined Post Architecture Cost drivers
RCPX	RELY, DATA, CPLX, DOCU
RUSE	RUSE
PDIF	TIME, STOR, PVOL
PERS	ACAP, PCAP, PCON
PREX	AEXP, PEXP, LTEX
FCIL	TOOL, SITE
SCED	SCED

Table 14: Mapping table

Software Project Planning

Product of cost drivers for early design model

i. Product Reliability and Complexity (RCPX): The cost driver combines four Post Architecture cost drivers which are RELY, DATA, CPLX and DOCU.

RCPX	*Extra Low*	*Very Low*	*Low*	*Nominal*	*High*	*Very High*	*Extra High*
Sum of RELY, DATA, CPLX, DOCU ratings	5, 6	7, 8	9-11	12	13-15	16-18	19-21
Emphasis on reliability, documentation	Very Little	Little	Some	Basic	Strong	Very Strong	Extreme
Product complexity	Very Simple	Simple	Some	Moderate	Complex	Very Complex	Extremely Complex
Database size	Small	Small	Small	Moderate	Large	Very Large	Very Large

Software Project Planning

ii. Required Reuse (RUSE) : This early design model cost driver is same as its Post architecture Counterpart. The RUSE rating levels are (as per Table 16):

	Vary Low	*Low*	*Nominal*	*High*	*Very High*	*Extra High*
RUSE	1	2	3	4	5	6
		None	Across project	Across program	Across product line	Across multiple product line

Software Project Planning

iii. Platform Difficulty (PDIF) : This cost driver combines TIME, STOR and PVOL of Post Architecture Cost Drivers.

PDIF	*Low*	*Nominal*	*High*	*Very High*	*Extra High*
Sum of Time, STOR & PVOL ratings	8	9	10-12	13-15	16-17
Time & storage constraint	≤ 50%	≤ 50%	65%	80%	90%
Platform Volatility	Very stable	Stable	Somewhat stable	Volatile	Highly Volatile

Software Project Planning

iv. Personnel Capability (PERS) : This cost driver combines three Post Architecture Cost Drivers. These drivers are ACAP, PCAP and PCON.

PERS	*Extra Low*	*Very Low*	*Low*	*Nominal*	*High*	*Very High*	*Extra High*
Sum of ACAP, PCAP, PCON ratings	3, 4	5, 6	7, 8	9	10, 11	12, 13	14, 15
Combined ACAP & PCAP Percentile	20%	39%	45%	55%	65%	75%	85%
Annual Personnel Turnover	45%	30%	20%	12%	9%	5%	4%

Software Project Planning

v. Personnel Experience (PREX) : This early design driver combines three Post Architecture Cost Drivers, which are AEXP, PEXP and LTEX.

PREX	*Extra Low*	*Very Low*	*Low*	*Nominal*	*High*	*Very High*	*Extra High*
Sum of AEXP, PEXP and LTEX ratings	3, 4	5, 6	7, 8	9	10, 11	12, 13	14, 15
Applications, Platform, Language & Tool Experience	≤ 3 months	5 months	9 months	1 year	2 year	4 year	6 year

Software Project Planning

vi. Facilities (FCIL): This depends on two Post Architecture Cost Drivers, which are TOOL and SITE.

FCIL	*Extra Low*	*Very Low*	*Low*	*Nominal*	*High*	*Very High*	*Extra High*
Sum of TOOL & SITE ratings	2	3	4, 5	6	7, 8	9, 10	11
Tool support	Minimal	Some	Simple CASE tools	Basic life cycle tools	Good support of tools	Very strong use of tools	Very strong & well integrated tools
Multisite conditions development support	Weak support of complex multisite development	Some support	Moderate support	Basic support	Strong support	Very strong support	Very strong support

Software Project Planning

vii. Schedule (SCED) : This early design cost driver is the same as Post Architecture Counterpart and rating level are given below using table 16.

SCED	*Very Low*	*Low*	*Nominal*	*High*	*Very High*
Schedule	75% of Nominal	85%	100%	130%	160%

Software Project Planning

The seven early design cost drivers have been converted into numeric values with a Nominal value 1.0. These values are used for the calculation of a factor called "Effort multiplier" which is the product of all seven early design cost drivers. The numeric values are given in Table 15.

Early design Cost drivers	*Extra Low*	*Very Low*	*Low*	*Nominal*	*High*	*Very High*	*Extra High*
RCPX	.73	.81	.98	1.0	1.30	1.74	2.38
RUSE	—	—	0.95	1.0	1.07	1.15	1.24
PDIF	—	—	0.87	1.0	1.29	1.81	2.61
PERS	2.12	1.62	1.26	1.0	0.83	0.63	0.50
PREX	1.59	1.33	1.12	1.0	0.87	0.71	0.62
FCIL	1.43	1.30	1.10	1.0	0.87	0.73	0.62
SCED	—	1.43	1.14	1.0	1.0	1.0	—

Table 15: Early design parameters

Software Project Planning

The early design model adjusts the nominal effort using 7 effort multipliers (EMs). Each effort multiplier (also called drivers) has 7 possible weights as given in Table 15. These factors are used for the calculation of adjusted effort as given below:

$$PM_{adjusted} = PM_{\text{nominal}} \times \left[\prod_{i=7}^{7} EM_i \right]$$

$PM_{adjusted}$ effort may very even up to 400% from $PM_{nominal}$

Hence $PM_{adjusted}$ is the fine tuned value of effort in the early design phase

Software Project Planning

Example: 4.10

A software project of application generator category with estimated 50 KLOC has to be developed. The scale factor (B) has low precedentness, high development flexibility and low team cohesion. Other factors are nominal. The early design cost drivers like platform difficult (PDIF) and Personnel Capability (PERS) are high and others are nominal. Calculate the effort in person months for the development of the project.

Software Project Planning

Solution

Here B = 0.91 + 0.01 * (Sum of rating on scaling factors for the project)

= 0.91 + 0.01 * (4.96 + 2.03 + 4.24 + 4.38 + 4.68)

= 0.91 + 0.01(20.29)=1.1129

$PM_{nominal} = A*(size)^{B}$

$= 2.5 * (50)^{1.1129} = 194.41$ Person months

The 7 cost drivers are

PDIF = high (1.29)

PERS = high (0.83)

RCPX = nominal (1.0)

RUSE = nominal (1.0)

PREX = nominal (1.0)

FCIL = nominal (1.0)

SCEO = nominal (1.0)

Software Project Planning

$$PM_{adjusted} = PM_{\text{nominal}} \times \left[\prod_{i=7}^{7} EM_i \right]$$

= 194.41 * [1.29 x 0.83)

= 194.41 x 1.07

= 208.155 Person months

Post Architecture Model

The post architecture model is the most detailed estimation model and is intended to be used when a software life cycle architecture has been completed. This model is used in the development and maintenance of software products in the application generators, system integration or infrastructure sectors.

$$PM_{adjusted} = PM_{\text{nominal}} \times \left[\prod_{i=7}^{17} EM_i \right]$$

EM : Effort multiplier which is the product of 17 cost drivers.

The 17 cost drivers of the Post Architecture model are described in the table 16.

Software Project Planning

Cost driver	*Purpose*	*Very low*	*Low*	*Nominal*	*High*	*Very High*	*Extra High*
RELY (Reliability required)	Measure of the extent to which the software must perform its intended function over a period of time	Only slight inconvenience	Low, easily recoverable losses	Moderate, easily recoverable losses	High financial loss	Risk to human life	—
DATA (Data base size)	Measure the affect of large data requirements on product development	—	$\frac{\text{Database size(D)}}{\text{Prog. size (P)}} < 10$	$10 \le \frac{D}{P} < 100$	$100 \le \frac{D}{P} < 1000$	$\frac{D}{P} \ge 1000$	—
CPLX (Product complexity)	Complexity is divided into five areas: Control operations, computational operations, device dependent operations, data management operations & User Interface management operations.	See Table 4.17					
DOCU Documentation	Suitability of the project's documentation to its life cycle needs	Many life cycle needs uncovered	Some needs uncovered	Adequate	Excessive for life cycle needs	Very Excessive	—

Table 16: Post Architecture Cost Driver rating level summary

Cont…

Software Project Planning

TIME (Execution Time constraint)	Measure of execution time constraint on software	—	—	≤ 50% use of a available execution time	70%	85%	95%
STOR (Main storage constraint)	Measure of main storage constraint on software	—	—	≤ 50% use of available storage	70%	85%	95%
PVOL (Platform Volatility)	Measure of changes to the OS, compilers, editors, DBMS etc.	—	Major changes every 12 months & minor changes every 1 month	Major: 6 months Minor: 2 weeks	Major: 2 months Minor: 1 week	Major: 2 week Minor: 2 days	—
ACAP (Analyst capability)	Should include analysis and design ability, efficiency & thoroughness, and communication skills.	15th Percentile	35th Percentile	55th Percentile	75th Percentile	90th Percentile	—

Table 16: Post Architecture Cost Driver rating level summary

Cont…

Software Project Planning

PCAP (Programmers capability)	Capability of Programmers as a team. It includes ability, efficiency, thoroughness & communication skills	15th Percentile	35th Percentile	55th Percentile	75th Percentile	90th Percentile	—
PCON (Personnel Continuity)	Rating is in terms of Project's annual personnel turnover	48%/year	24%/year	12%/year	6%/year	3%/year	—
AEXP (Applications Experience)	Rating is dependent on level of applications experience.	≤ 2 months	6 months	1 year	3 year	6 year	—
PEXP (Platform experience)	Measure of Platform experience	≤ 2 months	6 months	1 year	3 year	6 year	—

Table 16: Post Architecture Cost Driver rating level summary

Cont…

Software Project Planning

LTEX (Language & Tool experience)	Rating is for Language & tool experience	≤ 2 months	6 months	1 year	3 year	6 year	—
TOOL (Use of software tools)	It is the indicator of usage of software tools	No use	Beginning to use	Some use	Good use	Routine & habitual use	—
SITE (Multisite development)	Site location & Communication technology between sites	International with some phone & mail facility	Multicity & multi company with individual phones, FAX	Multicity & multi company with Narrow band mail	Same city or Metro with wideband electronic communication	Same building or complex with wideband electronic communication & Video conferencing	Fully co-located with interactive multimedia
SCED (Required Development Schedule)	Measure of Schedule constraint. Ratings are defined in terms of percentage of schedule stretch-out or acceleration with respect to nominal schedule	75% of nominal	85%	100%	130%	160%	—

Table 16: Post Architecture Cost Driver rating level summary

Software Project Planning

Product complexity is based on control operations, computational operations, device dependent operations, data management operations and user interface management operations. Module complexity rating are given in table 17.

The numeric values of these 17 cost drivers are given in table 18 for the calculation of the product of efforts i.e., effort multiplier (EM). Hence PM adjusted is calculated which will be a better and fine tuned value of effort in person months.

Software Project Planning

	Control Operations	Computational Operations	Device-dependent Operations	Data management Operations	User Interface Management Operations
Very Low	Straight-line code with a few non-nested structured programming operators: Dos. Simple module composition via procedure calls or simple scripts.	Evaluation of simple expressions: e.g., A=B+C*(D-E)	Simple read, write statements with simple formats.	Simple arrays in main memory. Simple COTSDB queries, updates.	Simple input forms, report generators.
Low	Straight forward nesting of structured programming operators. Mostly simple predicates	Evaluation of moderate-level expressions: e.g., D=SQRT(B**2-4*A*C)	No cognizance needed of particular processor or I/O device characteristics. I/O done at GET/PUT level.	Single file sub setting with no data structure changes, no edits, no intermediate files, Moderately complex COTS-DB queries, updates.	User of simple graphics user interface (GUI) builders.

Table 17: Module complexity ratings

Cont…

Software Project Planning

	Control Operations	Computational Operations	Device-dependent Operations	Data management Operations	User Interface Management Operations
Nominal	Mostly simple nesting. Some inter module control Decision tables. Simple callbacks or message passing, including middleware supported distributed processing.	Use of standard maths and statistical routines. Basic matrix/ vector operations.	I/O processing includes device selection, status checking and error processing.	Multi-file input and single file output. Simple structural changes, simple edits. Complex COTS-DB queries, updates.	Simple use of widget set.
High	Highly nested structured programming operators with many compound predicates. Queue and stack control. Homogeneous, distributed processing. Single processor soft real time control.	Basic numerical analysis: multivariate interpolation, ordinary differential equations. Basic truncation, round off concerns.	Operations at physical I/O level (physical storage address translations; seeks, read etc.) Optimized I/O overlap.	Simple triggers activated by data stream contents. Complex data restructuring.	Widget set development and extension. Simple voice I/O multimedia.

Table 17: Module complexity ratings

Cont…

Software Project Planning

	Control Operations	Computational Operations	Device-dependent Operations	Data management Operations	User Interface Management Operations
Very High	Reentrant and recursive coding. Fixed-priority interrupt handling. Task synchronization, complex callbacks, heterogeneous distributed processing. Single processor hard real time control.	Difficult but structured numerical analysis: near singular matrix equations, partial differential equations. Simple parallelization.	Routines for interrupt diagnosis, servicing, masking. Communication line handling. Performance intensive embedded systems.	Distributed database coordination. Complex triggers. Search optimization.	Moderately complex 2D/3D, dynamic graphics, multimedia.
Extra High	Multiple resource scheduling with dynamically changing priorities. Microcode-level control. Distributed hard real time control.	Difficult and unstructured numerical analysis: highly accurate analysis of noisy, stochastic data. Complex parallelization.	Device timing dependent coding, micro programmed operations. Performance critical embedded systems.	Highly coupled, dynamic relational and object structures. Natural language data management.	Complex multimedia, virtual reality.

Table 17: Module complexity ratings

Software Project Planning

Cost Driver	Rating					
	Very Low	Low	Nominal	High	Very High	Extra High
RELY	0.75	0.88	1.00	1.15	1.39	
DATA		0.93	1.00	1.09	1.19	
CPLX	0.75	0.88	1.00	1.15	1.30	1.66
RUSE		0.91	1.00	1.14	1.29	1.49
DOCU	0.89	0.95	1.00	1.06	1.13	
TIME			1.00	1.11	1.31	1.67
STOR			1.00	1.06	1.21	1.57
PVOL		0.87	1.00	1.15	1.30	
ACAP	1.50	1.22	1.00	0.83	0.67	
PCAP	1.37	1.16	1.00	0.87	0.74	

Table 18: 17 Cost Drivers

Cont…

Software Project Planning

Cost Driver	Rating					
	Very Low	Low	Nominal	High	Very High	Extra High
PCON	1.24	1.10	1.00	0.92	0.84	
AEXP	1.22	1.10	1.00	0.89	0.81	
PEXP	1.25	1.12	1.00	0.88	0.81	
LTEX	1.22	1.10	1.00	0.91	0.84	
TOOL	1.24	1.12	1.00	0.86	0.72	
SITE	1.25	1.10	1.00	0.92	0.84	0.78
SCED	1.29	1.10	1.00	1.00	1.00	

Table 18: 17 Cost Drivers

Software Project Planning

Schedule estimation

Development time can be calculated using $PM_{adjusted}$ as a key factor and the desired equation is:

$$TDEV_{\text{nominal}} = [\phi \times (PM_{adjusted})^{(0.28+0.2(B-0.091))}] * \frac{SCED\%}{100}$$

where Φ = constant, provisionally set to 3.67

$TDEV_{nominal}$ = calendar time in months with a scheduled constraint

B = Scaling factor

$PM_{adjusted}$ = Estimated effort in Person months (after adjustment)

Software Project Planning

Size measurement

Size can be measured in any unit and the model can be calibrated accordingly. However, COCOMO II details are:

i. Application composition model uses the size in object points.

ii. The other two models use size in KLOC

Early design model uses unadjusted function points. These function points are converted into KLOC using Table 19. Post architecture model may compute KLOC after defining LOC counting rules. If function points are used, then use unadjusted function points and convert it into KLOC using Table 19.

Software Project Planning

Language	SLOC/UFP
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

Table 19: Converting function points to lines of code

Cont...

Software Project Planning

Language	SLOC/UFP
ANSI Cobol 85	91
Fortan 77	105
Forth	64
Jovial	105
Lisp	64
Modula 2	80
Pascal	91
Prolog	64
Report Generator	80
Spreadsheet	6

Table 19: Converting function points to lines of code

Software Project Planning

Example: 4.11

Consider the software project given in example 4.10. Size and scale factor (B) are the same. The identified 17 Cost drivers are high reliability (RELY), very high database size (DATA), high execution time constraint (TIME), very high analyst capability (ACAP), high programmers capability (PCAP). The other cost drivers are nominal. Calculate the effort in Person-Months for the development of the project.

Software Project Planning

Solution

Here B = 1.1129

$PM_{nominal}$ = 194.41 Person-months

$$PM_{adjusted} = PM_{\text{nominal}} \times \left[\prod_{i=7}^{17} EM_i \right]$$

= 194.41 x (1.15 x 1.19 x 1.11 x 0.67 x 0.87)

= 194.41 x 0.885

= 172.05 Person-months

Putnam Resource Allocation Model

Norden of IBM

→ Rayleigh curve

→ Model for a range of hardware development projects.

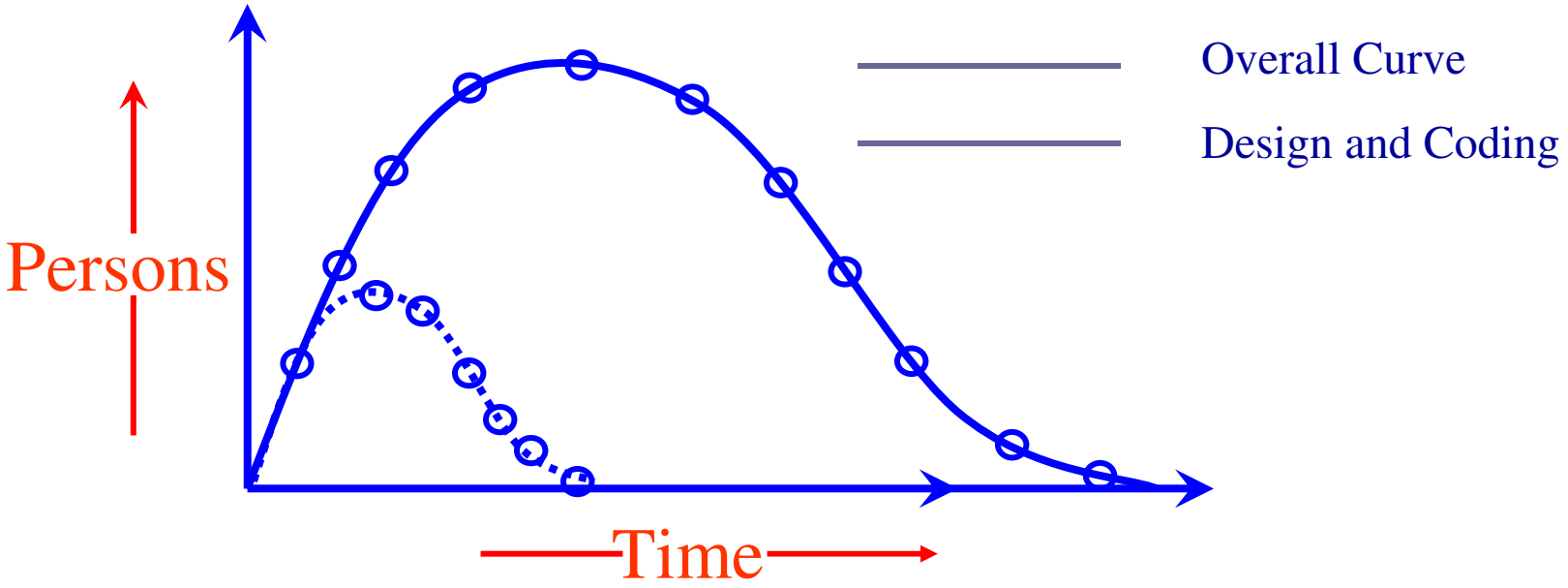


Fig.6: The Rayleigh manpower loading curve

Software Project Planning

Putnam observed that this curve was a close approximation at project level and software subsystem level.

No. of projects = 150

Software Project Planning

The Norden / Rayleigh Curve

The curve is modeled by differential equation

$$m(t) = \frac{dy}{dt} = 2kate^{-at^2} \qquad \text{--------- (1)}$$

$\frac{dy}{dt}$ = manpower utilization rate per unit time

a = parameter that affects the shape of the curve

K = area under curve in the interval [0, ∞]

t = elapsed time

Software Project Planning

On Integration on interval [o, t]

$$y(t) = K\,[1-e^{-at^2}] \quad \text{----------(2)}$$

Where y(t): cumulative manpower used upto time t.

$$y(0) = 0$$

$$y(\infty) = k$$

The cumulative manpower is null at the start of the project, and grows monotonically towards the total effort K (area under the curve).

Software Project Planning

$$\frac{d^2y}{dt^2} = 2kae^{-at^2}[1-2at^2] = 0$$

$$t_d^2 = \frac{1}{2a}$$

"t_d": time where maximum effort rate occurs

Replace "t_d" for t in equation (2)

$$E = y(t) = k\left(1 - e^{\frac{t_d^2}{2t_d^2}}\right) = K(1 - e^{-0.5})$$

$$E = y(t) = 0.3935\ k$$

$$a = \frac{1}{2t_d^2}$$

Software Project Planning

Replace "a" with $\frac{1}{2t_d^2}$ in the Norden/Rayleigh model. By making this substitution in equation we have

$$m(t) = \frac{2K}{2t_d^2} te^{-\frac{t^2}{2t_d^2}}$$

$$= \frac{K}{t_d^2} te^{-\frac{t^2}{2t_d^2}}$$

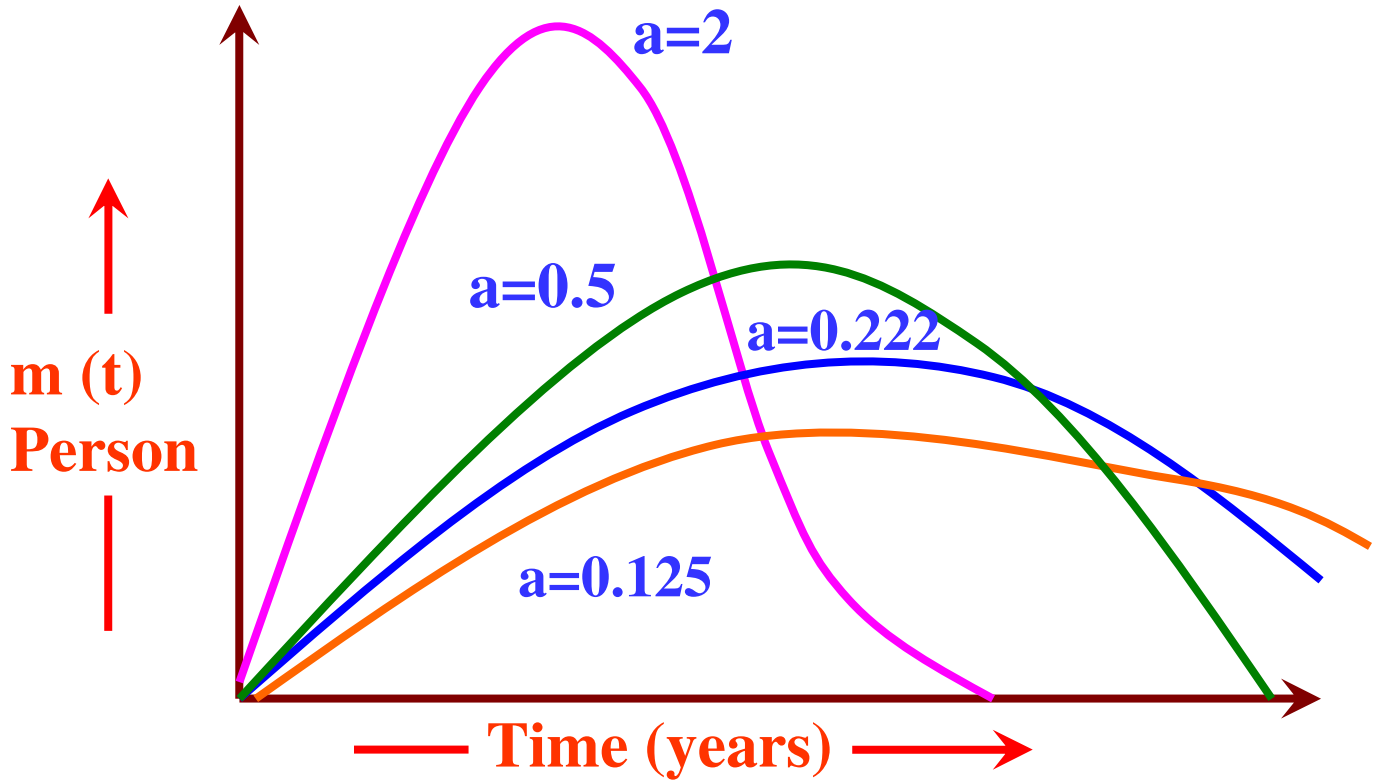


Fig.7: Influence of parameter 'a' on the manpower distribution

Software Project Planning

At time $t=t_d$, peak manning m (t_d) is obtained and denoted by m_o.

$$m_o = \frac{k}{t_d \sqrt{e}}$$

k = Total project cost/effort in person-years.

t_d = Delivery time in years

m_0 = No. of persons employed at the peak

e = 2.71828

Example: 4.12

A software development project is planned to cost 95 MY in a period of 1 year and 9 months. Calculate the peak manning and average rate of software team build up.

Software Project Planning

Solution

Software development cost k=95 MY

Peak development time $t_d = 1.75$ years

Peak manning $m_o = \frac{k}{t_d \sqrt{e}}$

$$\frac{95}{1.75 \times 1.648} = 32.94 = 33 \ persons$$

Average rate of software team build up

$$= \frac{m_0}{t_d} = \frac{33}{1.75} = 18.8 \, persons / \ year \ or \ 1.56 \ person / month$$

Example: 4.13

Consider a large-scale project for which the manpower requirement is K=600 PY and the development time is 3 years 6 months.

(a)Calculate the peak manning and peak time.

(b)What is the manpower cost after 1 year and 2 months?

Software Project Planning

Solution

(a) We know t_d=3 years and 6 months = 3.5 years

NOW $$m_0 = \frac{K}{t_d \sqrt{e}}$$

$$\therefore \quad m_0 = 600/(3.5 \text{x} 1.648) \cong 104 \text{ persons}$$

Software Project Planning

(b) We know

$$y(t) = K\left[1 - e^{-at^2}\right]$$

t = 1 year and 2 months

= 1.17 years

$$a = \frac{1}{2t_d^2} = \frac{1}{2\times(3.5)^2} = 0.041$$

$$y(1.17) = 600\left[1 - e^{-0.041\,(1.17)^2}\right]$$

= 32.6 PY

Software Project Planning

Difficulty Metric

Slope of manpower distribution curve at start time t=0 has some useful properties.

$$m'(t) = \frac{d^2 y}{dt^2} = 2kae^{-at^2}(1 - 2at^2)$$

Then, for t=0

$$m'(0) = 2Ka = \frac{2K}{2t_d^2} = \frac{K}{t_d^2}$$

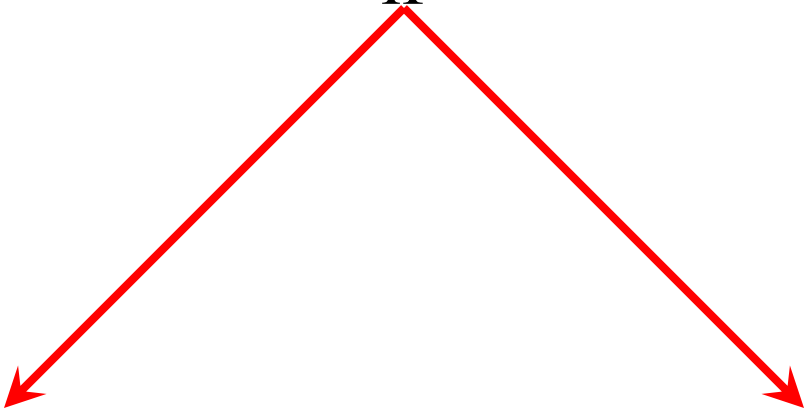
Software Project Planning

The ratio $\frac{K}{t_d^2}$ is called difficulty and denoted by D, which is measured in person/year :

$$D = \frac{k}{t_d^2} \text{ persons/year}$$

Software Project Planning

Project is difficult to develop if



Manpower demand is high

When time schedule is short

Software Project Planning

Peak manning is defined as:

$$m_0 = \frac{k}{t_d \sqrt{e}}$$

$$D = \frac{k}{t_d^2} = \frac{m_0 \sqrt{e}}{t_d}$$

Thus difficult projects tend to have a higher peak manning for a given development time, which is in line with Norden's observations relative to the parameter "a".

Software Project Planning

Manpower buildup

D is dependent upon "K". The derivative of D relative to "K" and "t_d" are

$$D'(t_d) = \frac{-2k}{t_d^3} \; persons / \; year^2$$

$$D'(k) = \frac{1}{t_d^2} \; year^{-2}$$

Software Project Planning

$D^1(K)$ will always be very much smaller than the absolute value of $D^1(t_d)$. This difference in sensitivity is shown by considering two projects

Project A : Cost = 20 PY & t_d = 1 year
Project B : Cost = 120 PY & t_d = 2.5 years

The derivative values are

Project A : $D`(t_d) = -40$ & $D`(K) = 1$
Project B : $D`(t_d) = -15.36$ & $D`(K) = 0.16$

This shows that a given software development is time sensitive.

Software Project Planning

Putnam observed that

Difficulty derivative relative to time

Behavior of s/w development

If project scale is increased, the development time also increase to such an extent that $\frac{k}{t_d^3}$ remains constant

around a value which could be 8,15,27.

Software Project Planning

It is represented by D_0 and can be expressed as:

$$D_0 = \frac{k}{t_d^3} \; person / \; year^2$$

D_0 =8, new s/w with many interfaces & interactions with other systems.

D_0 =15, New standalone system.

D_0 =27, The software is rebuild form existing software.

Software Project Planning

Example: 4.14

Consider the example 4.13 and calculate the difficulty and manpower build up.

Software Project Planning

Solution

We know

Difficulty $$D = \frac{K}{t_d^2}$$

$$= \frac{600}{(3.5)^2} = 49 \ person/year$$

Manpower build up can be calculated by following equation

$$D_0 = \frac{K}{t_d^3}$$

$$= \frac{600}{(3.5)^3} = 14 \ person/year^2$$

Productivity Versus Difficulty

Productivity = No. of LOC developed per person-month

$$P \propto D^{\beta}$$

Avg. productivity

$$P = \frac{LOC\ produced}{cumulative\ manpower\ used\ to\ produce\ code}$$

Software Project Planning

P = S/E

$$P = \phi D^{-2/3}$$

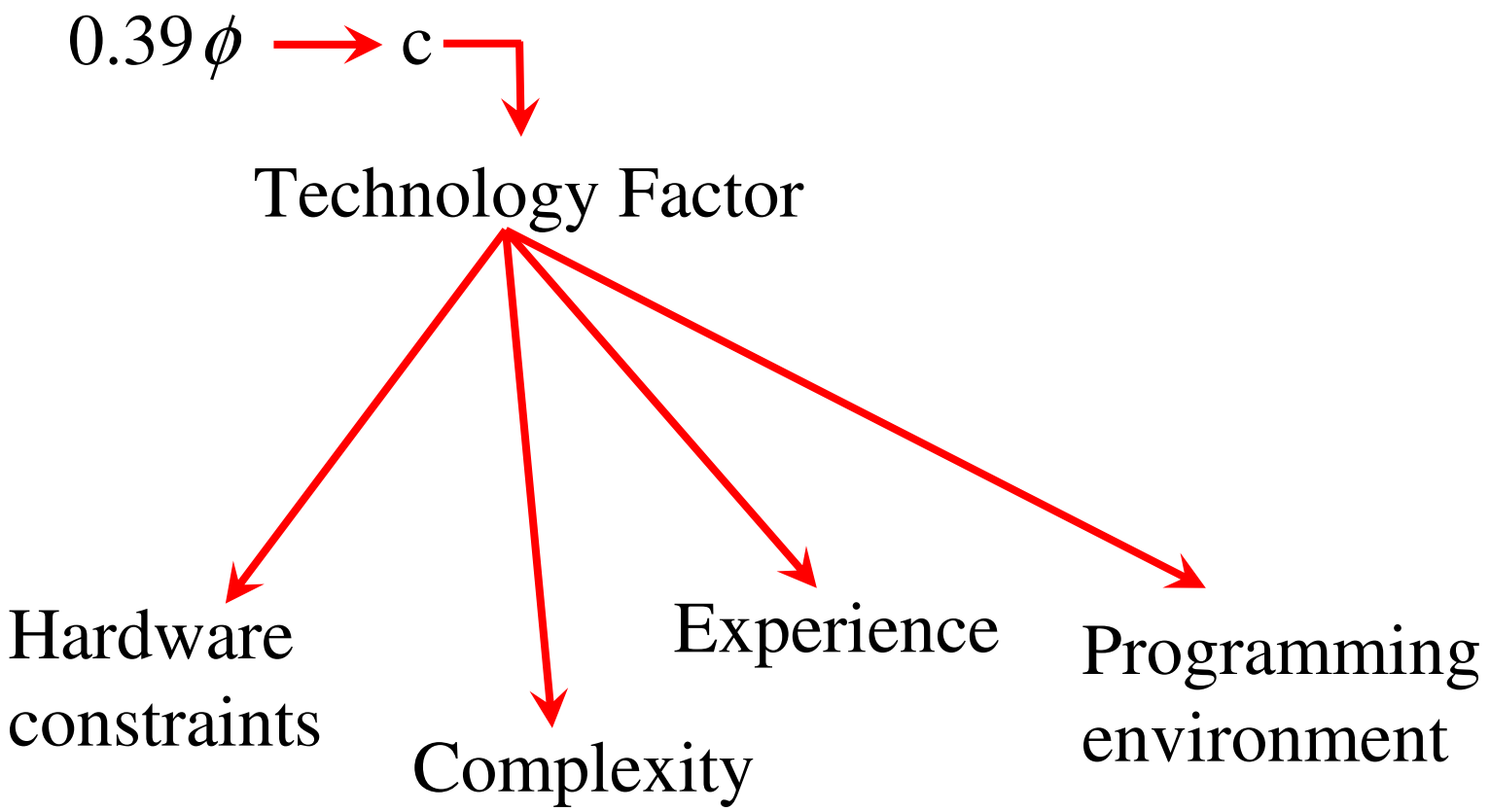
$$S = \phi D^{-2/3} E$$

$$= \phi D^{-2/3}(0.3935\ K)$$

$$S = \phi \left[\frac{k}{t_d^2}\right]^{-\frac{2}{3}} k(0.3935)$$

$$S = 0.3935\,\phi\, K^{1/3} t_d^{\,4/3}$$

Software Project Planning



Software Project Planning

C → 610 – 57314

K : P-Y

T : Years

$$S = CK^{1/3} t_d^{4/3}$$

$$C = S.K^{-1/3} t_d^{-4/3}$$

The trade off of time versus cost

$$K^{1/3} t_d^{4/3} = S / C$$

$$K = \frac{1}{t_d^4} \left(\frac{S}{C} \right)^3$$

Software Project Planning

C = 5000

S = 5,00,000 LOC

$$K = \frac{1}{t_d^4}(100)^3$$

t_d (years)	K (P-Y)
5.0	1600
4.0	3906
3.5	6664
3.0	12346

Table 20: (Manpower versus development time)

Software Project Planning

Development Subcycle

All that has been discussed so far is related to project life cycle as represented by project curve

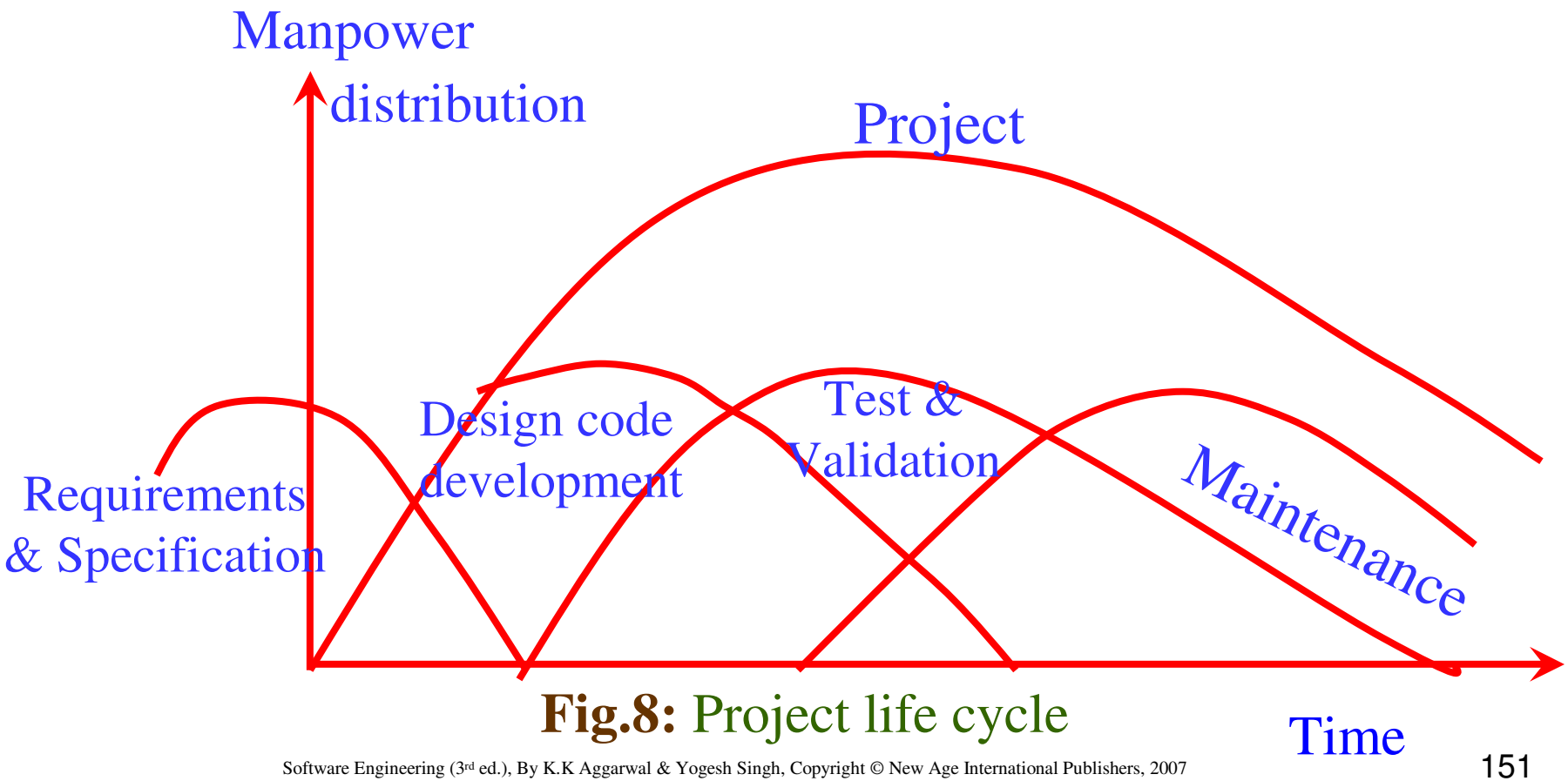
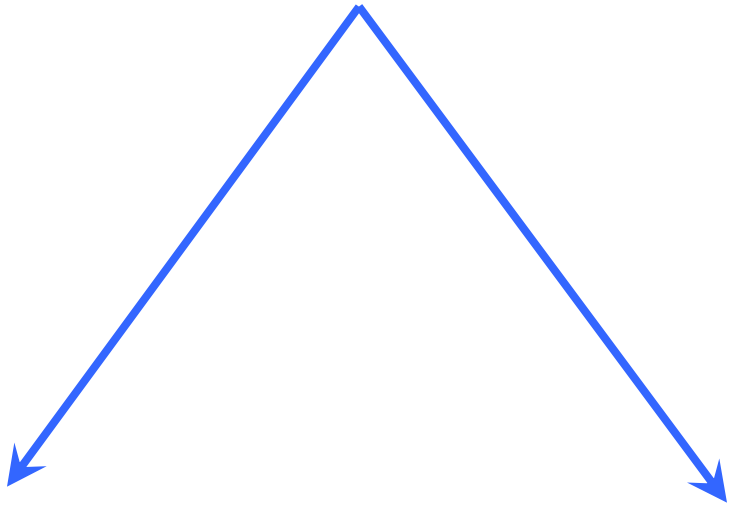


Fig.8: Project life cycle

Software Project Planning

Project life cycle

Project curve is the addition of two curves



Development Curve

Test & Validation Curve

Software Project Planning

$$\therefore\ m_d(t) = 2k_d bt\, e^{-bt^2}$$

$$y_d(t) = K_d\,[1-e^{-bt^2}]$$

An examination of $m_d(t)$ function shows a non-zero value of m_d at time t_d.

This is because the manpower involved in design & coding is still completing this activity after t^d in form of rework due to the validation of the product.

Nevertheless, for the model, a level of completion has to be assumed for development.

It is assumed that 95% of the development will be completed by the time t_d.

Software Project Planning

$$\frac{y_d(t)}{K_d} = 1 - e^{-bt^2} = 0.95$$

$\therefore$ We may say that $b = \frac{1}{2t_{od}^2}$

T_{od}: time at which development curve exhibits a peak manning.

$$t_{od} = \frac{t_d}{\sqrt{6}}$$

Software Project Planning

Relationship between K_d & K must be established.

At the time of origin, both cycles have the same slope.

$$\left(\frac{dm}{dt}\right)_o = \frac{K}{t_d^2} = \frac{K_d}{t_{od}^2} = \left(\frac{dm_d}{dt}\right)_o$$

K_d=K/6

$$D = \frac{K}{t_d^2} = \frac{K_d}{t_{od}^2}$$

Software Project Planning

This does not apply to the manpower build up D_0.

$$D_o = \frac{K}{t_d^3} = \frac{K_d}{\sqrt{6}t_{od}^3}$$

Conte investigated that

Larger projects ⟶ reasonable

Medium & small projects ⟶ overestimate

Example: 4.15

A software development requires 90 PY during the total development sub-cycle. The development time is planned for a duration of 3 years and 5 months

(a) Calculate the manpower cost expended until development time

(b) Determine the development peak time

(c) Calculate the difficulty and manpower build up.

Software Project Planning

Solution

(a) Duration t_d = 3.41 years

We know from equation $\frac{y_d(t)}{K_d} = 1 - e^{-bt_d} = 0.95$

$$\frac{y_d(t_d)}{K_d} = 0.95$$

$$Y_d(t_d) = 0.95 \times 90$$

= 85.5 PY

Software Project Planning

(b) We know from equation $t_{od} = \frac{t_d}{\sqrt{6}}$

$$t_{od} = \frac{t_d}{\sqrt{6}} = 3.41 / 2.449 = 1.39 \ years$$

$$\cong 17 \ months$$

Software Project Planning

(c) Total Manpower development

$$K_d = y_d(t_d) / 0.95$$

$$= 85.5 / 0.95 = 90$$

$$K = 6K_d = 90 \times 6 = 540 PY$$

$$D = K / t_d^2 = 540/(3.41)^2 = 46 \text{ persons/years}$$

$$D_o = \frac{K}{t_d^3} = 540/(3.41)^3 = 13.6 \text{ persons/years}^2$$

Example:4.16

A software development for avionics has consumed 32 PY up to development cycle and produced a size of 48000 LOC. The development of project was completed in 25 months. Calculate the development time, total manpower requirement, development peak time, difficulty, manpower build up and technology factor.

Software Project Planning

Solution:

Development time t_d = 25 months = 2.08 years

Total manpower development $k_d = \frac{Y_d(t_d)}{0.95} = \frac{32}{0.95} = 33.7\ PY$

Development peak time $t_{od} = \frac{(t_d)}{\sqrt{6}} = 0.85\ years = 10\ months$

$$K = 6K_d = 6 \text{ x } 33.7 = 202 \text{ PY}$$

$$D = \frac{k}{t_d^2} = \frac{202}{(2.08)^2} = 46.7\ pesons / years$$

Software Project Planning

$$D_0 = \frac{k}{t_d^3} = \frac{202}{(2.08)^3} = 22.5 \; Persons / \; year^2$$

Technology factor

$$C = SK^{-1/3} t_d^{-4/3}$$

$$= 3077$$

Example 4.17

What amount of software can be delivered in 1 year 10 months in an organization whose technology factor is 2400 if a total of 25 PY is permitted for development effort.

Software Project Planning

Solution:

$t_d = 1.8$ years

$K_d = 25$ PY

$K = 25 \times 6 = 150$ PY

$C = 2400$

We know $S = CK^{1/3} t_d^{4/3}$

$= 2400 \times 5.313 \times 2.18 = 27920$ LOC

Software Project Planning

Example 4.18

The software development organization developing real time software has been assessed at technology factor of 2200. The maximum value of manpower build up for this type of software is D_o=7.5. The estimated size to be developed is S=55000 LOC.

(a) Determine the total development time, the total development manpower cost, the difficulty and the development peak manning.

(b) The development time determined in (a) is considered too long. It is recommended that it be reduced by two months. What would happen?

Solution

We have $$S = CK^{1/3} t_d^{4/3}$$

$$\left(\frac{s}{c}\right)^3 = kt_d^4$$

which is also equivalent to $$\left(\frac{S}{C}\right)^3 = D_o t_d^7$$

then $$t_d = \left[\frac{1}{D_0}\left(\frac{S}{C}\right)^3\right]^{1/7}$$

Software Project Planning

$$Since \quad \frac{S}{C} = 25$$

$t_d = 3$ years

$$K = D_0 t_d^3 = 7.5 \times 27 = 202\ PY$$

Total development manpower cost $K_d = \frac{202}{06} = 3375 PY$

$D = D_0 t_d = 22.5$ persons / year

$$t_{od} = \frac{t_d}{\sqrt{6}} = \frac{3}{\sqrt{6}} = 1.2\ years$$

Software Project Planning

$$M_d(t) = 2k_d\, bte^{-bt^2}$$

$$Y_d(t) = k_d\,(1\text{-}e^{-bt^2})$$

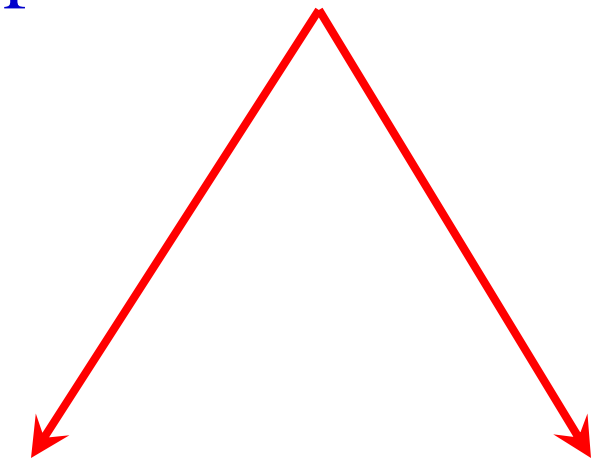
Here $t = t_{od}$

Peak manning $= m_{od} = Dt_{od}e^{-1/2}$

$= 22.5 \times 1.2 \times .606 = 16$ persons

Software Project Planning

III. If development time is reduced by 2 months



Developing s/w at higher manpower build-up

Producing less software

Software Project Planning

(i) Increase Manpower Build-up

$$D_o = \frac{1}{t_d^7}\left(\frac{S}{C}\right)^3$$

Now $t_d = 3$ years $- 2$ months $= 2.8$ years

$$D_o = (25)^3 / (2.8)^7 = 11.6 \; persons / years$$

$$k = D_0 t_d^3 = 254 \, PY$$

$$K_d = \frac{254}{6} = 42.4 \, PY$$

Software Project Planning

$$D = D_0 t_d = 32.5 \text{ persons / year}$$

The peak time is $t_{od} = 1.14$ years

Peak manning $m_{od} = Dt_{od} e^{-0.5}$

$= 32.5 \times 1.14 \times 0.6$

$= 22$ persons

Note the huge increase in peak manning & manpower cost.

Software Project Planning

(ii) Produce Less Software

$$\left(\frac{S}{C}\right)^3 = D_0 t_d^7 = 7.5 \times (2.8)^7 = 10119.696$$

$$\left(\frac{S}{C}\right)^3 = 21.62989$$

Then for C=2200

S=47586 LOC

Productivity versus difficult

Example 4.19

A stand alone project for which the size is estimated at 12500 LOC is to be developed in an environment such that the technology factor is 1200. Choosing a manpower build up D_o=15, Calculate the minimum development time, total development man power cost, the difficulty, the peak manning, the development peak time, and the development productivity.

Software Project Planning

Solution

Size (S) = 12500 LOC

Technology factor (C) = 1200

Manpower buildup (D_o) = 15

Now

$$S = CK^{1/3}t_d^{4/3}$$

$$\frac{S}{C} = K^{1/3}t_d^{4/3}$$

$$\left(\frac{S}{C}\right)^3 = Kt_d^4$$

Software Project Planning

Also we know $D_o = \frac{K}{t_d^3}$

$$K = D_o t_d^3 = D_o t_d^3$$

Hence $\left(\frac{S}{C}\right)^3 = D_o t_d^7$

Substituting the values, we get $\left(\frac{12500}{1200}\right)^3 = 15 t_d^7$

$$t_d = \left[\frac{(10.416)^3}{15}\right]^{1/7}$$

$$t_d = 1.85 \text{ years}$$

Software Project Planning

(i) Hence Minimum development time (t_d)=1.85 years

(ii) Total development manpower cost $K_d = \frac{K}{6}$

Hence $K = 15 t_d^3$

$= 15(1.85)^3 = 94.97$ PY

$$K_d = \frac{K}{6} = \frac{94.97}{6} = 15.83\ PY$$

(iii) Difficulty $$D = \frac{K}{t_d^2} = \frac{94.97}{(1.85)^2} = 27.75\ Persons / year$$

Software Project Planning

(iv) Peak Manning $$m_0 = \frac{K}{t_d \sqrt{e}}$$

$$= \frac{94.97}{1.85 \times 1.648} = 31.15 Person$$

(v) Development Peak time $$t_{od} = \frac{t_d}{\sqrt{6}}$$

$$= \frac{1.85}{2.449} = 0.755 \ years$$

Software Project Planning

(vi) Development Productivity

$$= \frac{No\ .of\ \ lines\ \ of\ \ code\ \ (S)}{effort\ \ (K_d)}$$

$$= \frac{12500}{15.83} = 789.6\ LOC / PY$$

Software Project Planning

Software Risk Management

- We Software developers are extremely optimists.
- We assume, everything will go exactly as planned.
- Other view
  - not possible to predict what is going to happen ?

Software surprises

- Never good news

Software Project Planning

Risk management is required to reduce this surprise factor

Dealing with concern before it becomes a crisis.

Quantify probability of failure & consequences of failure.

What is risk ?

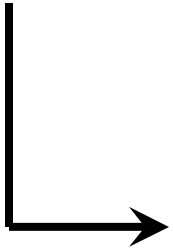
Tomorrow's problems are today's risks.

"Risk is a problem that may cause some loss or threaten the success of the project, but which has not happened yet".

Software Project Planning

Risk management is the process of identifying addressing and eliminating these problems before they can damage the project.

Current problems &



Potential Problems

Software Project Planning

Typical Software Risk

Capers Jones has identified the top five risk factors that threaten projects in different applications.

1. Dependencies on outside agencies or factors.
   - Availability of trained, experienced persons
   - Inter group dependencies
   - Customer-Furnished items or information
   - Internal & external subcontractor relationships

Software Project Planning

2. Requirement issues

Uncertain requirements

Wrong product

or

Right product badly

Either situation results in unpleasant surprises and unhappy customers.

Software Project Planning

- Lack of clear product vision
- Lack of agreement on product requirements
- Unprioritized requirements
- New market with uncertain needs
- Rapidly changing requirements
- Inadequate Impact analysis of requirements changes

Software Project Planning

3. Management Issues

Project managers usually write the risk management plans, and most people do not wish to air their weaknesses in public.

- Inadequate planning
- Inadequate visibility into actual project status
- Unclear project ownership and decision making
- Staff personality conflicts
- Unrealistic expectation
- Poor communication

Software Project Planning

4. Lack of knowledge

- Inadequate training
- Poor understanding of methods, tools, and techniques
- Inadequate application domain experience
- New Technologies
- Ineffective, poorly documented or neglected processes

Software Project Planning

5. Other risk categories

- Unavailability of adequate testing facilities
- Turnover of essential personnel
- Unachievable performance requirements
- Technical approaches that may not work

Risk Management Activities

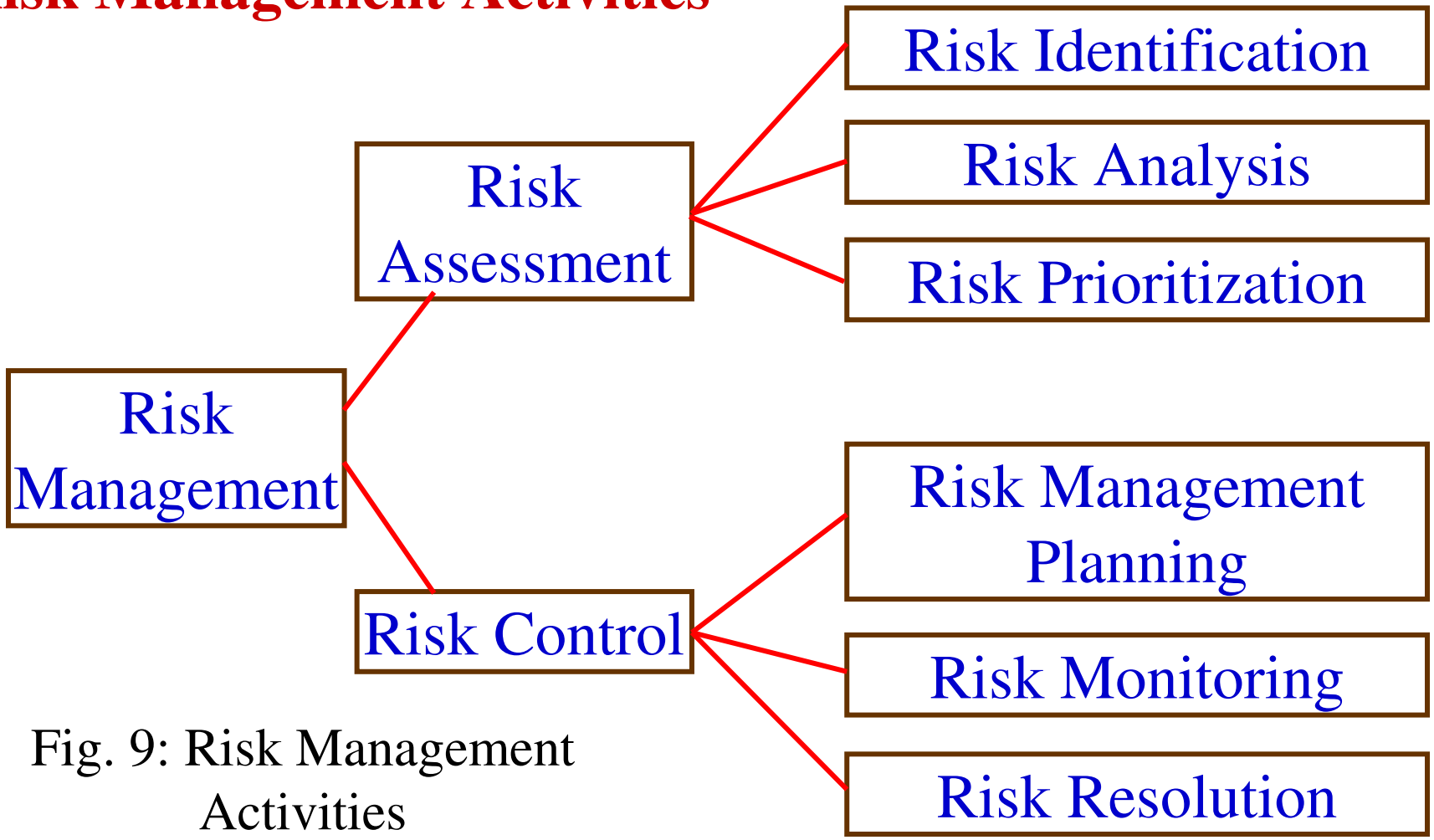


Fig. 9: Risk Management Activities

Risk Assessment

Identification of risks

Risk analysis involves examining how project outcomes might change with modification of risk input variables.

Risk prioritization focus for severe risks.

Risk exposure: It is the product of the probability of incurring a loss due to the risk and the potential magnitude of that loss.

Software Project Planning

Another way of handling risk is the risk avoidance. Do not do the risky things! We may avoid risks by not undertaking certain projects, or by relying on proven rather than cutting edge technologies.

Risk Control

Risk Management Planning produces a plan for dealing with each significant risks.

- Record decision in the plan.

Risk resolution is the execution of the plans of dealing with each risk.

Multiple Choice Questions

Note: Choose most appropriate answer of the following questions:

4.1 After the finalization of SRS, we may like to estimate

(a) Size
(b) Cost
(c) Development time
(d) All of the above.

4.2 Which one is not a size measure for software

(a) LOC
(b) Function Count
(c) Cyclomatic Complexity
(d) Halstead's program length

4.3 Function count method was developed by

(a) B.Beizer
(b) B.Boehm
(c) M.halstead
(d) Alan Albrecht

4.4 Function point analysis (FPA) method decomposes the system into functional units. The total number of functional units are

(a) 2
(b) 5
(c) 4
(d) 1

Multiple Choice Questions

4.5 IFPUG stand for
(a) Initial function point uniform group
(b) International function point uniform group
(c) International function point user group
(d) Initial function point user group

4.6 Function point can be calculated by
(a) UFP * CAF
(b) UFP * FAC
(c) UFP * Cost
(d) UFP * Productivity

4.7 Putnam resource allocation model is based on
(a) Function points
(b) Norden/ Rayleigh curve
(c) Putnam theory of software management
(d) Boehm's observation on manpower utilisation rate

4.8 Manpower buildup for Putnam resource allocation model is

$(a)\ K / t_d^2\ persons / year^2$
$(b)\ K / t_d^3\ persons / year^2$
$(c)\ K / t_d^2\ persons / year$
$(d)\ K / t_d^3\ persons / year$

Multiple Choice Questions

4.9 COCOMO was developed initially by

(a) B.W.Bohem
(b) Gregg Rothermal
(c) B.Beizer
(d) Rajiv Gupta

4.10 A COCOMO model is

(a) Common Cost estimation model
(b) Constructive cost Estimation model
(c) Complete cost estimation model
(d) Comprehensive Cost estimation model

4.11 Estimation of software development effort for organic software is COCOMO is

(a) $E=2.4(KLOC)^{1.05}PM$
(b) $E=3.4(KLOC)^{1.06}PM$
(c) $E=2.0(KLOC)^{1.05}PM$
(d) $E-2.4(KLOC)^{1.07}PM$

4.12 Estimation of size for a project is dependent on

(a) Cost
(b) Schedule
(c) Time
(d) None of the above

4.13 In function point analysis, number of Complexity adjustment factor are

(a) 10
(b) 20
(c) 14
(d) 12

Multiple Choice Questions

4.14 COCOMO-II estimation model is based on

(a) Complex approach
(b) Algorithm approach
(c) Bottom up approach
(d) Top down approach

4.15 Cost estimation for a project may include

(a) Software Cost
(b) Hardware Cost
(c) Personnel Costs
(d) All of the above

4.16 In COCOMO model, if project size is typically 2-50 KLOC, then which mode is to be selected?

(a) Organic
(b) Semidetached
(c) Embedded
(d) None of the above

4.17 COCOMO-II was developed at

(a) University of Maryland
(b) University of Southern California
(c) IBM
(d) AT & T Bell labs

4.18 Which one is not a Category of COCOMO-II

(a) End User Programming
(b) Infrastructure Sector
(c) Requirement Sector
(d) System Integration

Multiple Choice Questions

4.19 Which one is not an infrastructure software?

(a) Operating system
(b) Database management system
(c) Compilers
(d) Result management system

4.20 How many stages are in COCOMO-II?

(a) 2
(b) 3
(c) 4
(d) 5

4.21 Which one is not a stage of COCOMO-II?

(a) Application Composition estimation model
(b) Early design estimation model
(c) Post architecture estimation model
(d) Comprehensive cost estimation model

4.22 In Putnam resource allocation model, Rayleigh curve is modeled by the equation

$(a)\ \ m(t) = 2at\, e^{-at^2}$
$(b)\ \ m(t) = 2Kt\, e^{-at^2}$
$(c)\ \ m(t) = 2Kat\, e^{-at^2}$
$(d)\ \ m(t) = 2Kbt\, e^{-at^2}$

Multiple Choice Questions

4.23 In Putnam resource allocation model, technology factor 'C' is defined as

(a) $C = SK^{-1/3}t_d^{-4/3}$

(b) $C = SK^{1/3}t_d^{4/3}$

(c) $C = SK^{1/3}t_d^{-4/3}$

(d) $C = SK^{-1/3}t_d^{4/3}$

4.24 Risk management activities are divided in

(a) 3 Categories

(b) 2 Categories

(c) 5 Categories

(d) 10 Categories

4.25 Which one is not a risk management activity?

(a) Risk assessment

(b) Risk control

(c) Risk generation

(d) None of the above

Exercises

4.1 What are various activities during software project planning?

4.2 Describe any two software size estimation techniques.

4.3 A proposal is made to count the size of 'C' programs by number of semicolons, except those occurring with literal strings. Discuss the strengths and weaknesses to this size measure when compared with the lines of code count.

4.4 Design a LOC counter for counting LOC automatically. Is it language dependent? What are the limitations of such a counter?

4.5 Compute the function point value for a project with the following information domain characteristics.

Number of user inputs = 30

Number of user outputs = 42

Number of user enquiries = 08

Number of files = 07

Number of external interfaces = 6

Assume that all complexity adjustment values are moderate.

Exercises

4.6 Explain the concept of function points. Why FPs are becoming acceptable in industry?

4.7 What are the size metrics? How is function point metric advantageous over LOC metric? Explain.

4.8 Is it possible to estimate software size before coding? Justify your answer with suitable example.

4.9 Describe the Albrecht's function count method with a suitable example.

4.10 Compute the function point FP for a payroll program that reads a file of employee and a file of information for the current month and prints cheque for all the employees. The program is capable of handling an interactive command to print an individually requested cheque immediately.

Exercises

4.11 Assume that the previous payroll program is expected to read a file containing information about all the cheques that have been printed. The file is supposed to be printed and also used by the program next time it is run, to produce a report that compares payroll expenses of the current month with those of the previous month. Compute functions points for this program. Justify the difference between the function points of this program and previous one by considering how the complexity of the program is affected by adding the requirement of interfacing with another application (in this case, itself).

4.12 Explain the Walson & Felix model and compare with the SEL model.

4.13 The size of a software product to be developed has been estimated to be 22000 LOC. Predict the manpower cost (effort) by Walston-Felix Model and SEL model.

4.14 A database system is to be developed. The effort has been estimated to be 100 Persons-Months. Calculate the number of lines of code and productivity in LOC/Person-Month.

Exercises

4.15 Discuss various types of COCOMO mode. Explain the phase wise distribution of effort.

4.16 Explain all the levels of COCOMO model. Assume that the size of an organic software product has been estimated to be 32,000 lines of code. Determine the effort required to developed the software product and the nominal development time.

4.17 Using the basic COCOMO model, under all three operating modes, determine the performance relation for the ratio of delivered source code lines per person-month of effort. Determine the reasonableness of this relation for several types of software projects.

4.18 The effort distribution for a 240 KLOC organic mode software development project is: product design 12%, detailed design 24%, code and unit test 36%, integrate and test 28%. How would the following changes, from low to high, affect the phase distribution of effort and the total effort: analyst capability, use of modern programming languages, required reliability, requirements volatility?

Exercises

4.19 Specify, design, and develop a program that implements COCOMO. Using reference as a guide, extend the program so that it can be used as a planning tool.

4.20 Suppose a system for office automation is to be designed. It is clear from requirements that there will be five modules of size 0.5 KLOC, 1.5 KLOC, 2.0 KLOC, 1.0 KLOC and 2.0 KLOC respectively. Complexity, and reliability requirements are high. Programmer's capability and experience is low. All other factors are of nominal rating. Use COCOMO model to determine overall cost and schedule estimates. Also calculate the cost and schedule estimates for different phases.

4.21 Suppose that a project was estimated to be 600 KLOC. Calculate the effort and development time for each of the three modes i.e., organic, semidetached and embedded.

4.22 Explain the COCOMO-II in detail. What types of categories of projects are identified?

Exercises

4.23 Discuss the Infrastructure Sector of COCOMO-II.

4.24 Describe various stages of COCOMO-II. Which stage is more popular and why?

4.25 A software project of application generator category with estimated size of 100 KLOC has to be developed. The scale factor (B) has high percedentness, high development flexibility. Other factors are nominal. The cost drivers are high reliability, medium database size, high Personnel capability, high analyst capability. The other cost drivers are nominal. Calculate the effort in Person-Months for the development of the project.

4.26 Explain the Putnam resource allocation model. What are the limitations of this model?

4.27 Describe the trade-off between time versus cost in Putnam resource allocation model.

4.28 Discuss the Putnam resources allocation model. Derive the time and effort equations.

Exercises

4.29 Assuming the Putnam model, with S=100,000 , C=5000, D_o=15, Compute development time t_d and manpower development K_d.

4.30 Obtain software productivity data for two or three software development programs. Use several cost estimating models discussed in this chapter. How to the results compare with actual project results?

4.31 It seems odd that cost and size estimates are developed during software project planning-before detailed software requirements analysis or design has been conducted. Why do we think this is done? Are there circumstances when it should not be done?

4.32 Discuss typical software risks. How staff turnover problem affects software projects?

4.33 What are risk management activities? Is it possible to prioritize the risk?

Exercises

4.34 What is risk exposure? What techniques can be used to control each risk?

4.35 What is risk? Is it economical to do risk management? What is the effect of this activity on the overall cost of the project?

4.36 There are significant risks even in student projects. Analyze a student project and list all the risk.