Calculation and Estimation of EDPS System Design Time in Days

TOSHIO AWANO*

1. *Introduction*

A method of calculation and estimation of a term of EDPS system design is described in 3 parts.

(1) If all the elements are decided and the procedure is fixed, EDPS system design time is able to be calculated theoretically by setting up its PERT network.

(2) The estimation of programming days is calculated by introducing the 'risk' concept.

(3) The feedback times as Fig. 1 heve to be minimized, also, the most suitable times of amendments and re-trials must be estimated.

2. *Calculation and estimation of EDPS system design time*

The EDPS system design progress using an electronic computer is shown in in Fig. 1. There is a method of system design involving analyzing Fig. 1 in

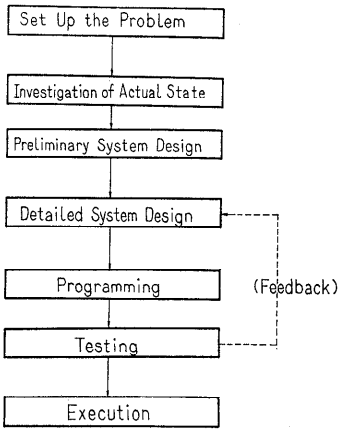

Fig. 1. EDPS progress.

detail [1]. Fig. 2 is its construction. Now, if the PERT network is able to be made by using this formulized system design construction and estimated working time for each activity is able to be written, the most suitable EDPS system design time can easily be calculated by its critical path [2] as Fig. 3.

This paper first appeared in Japanese in Joho Shori (the Journal of the Information Processing Society of Japan), Vol. 12, No. 5 (1971), pp. 272-279.

* Nippon Telegraph and Telephone Public Corporation.

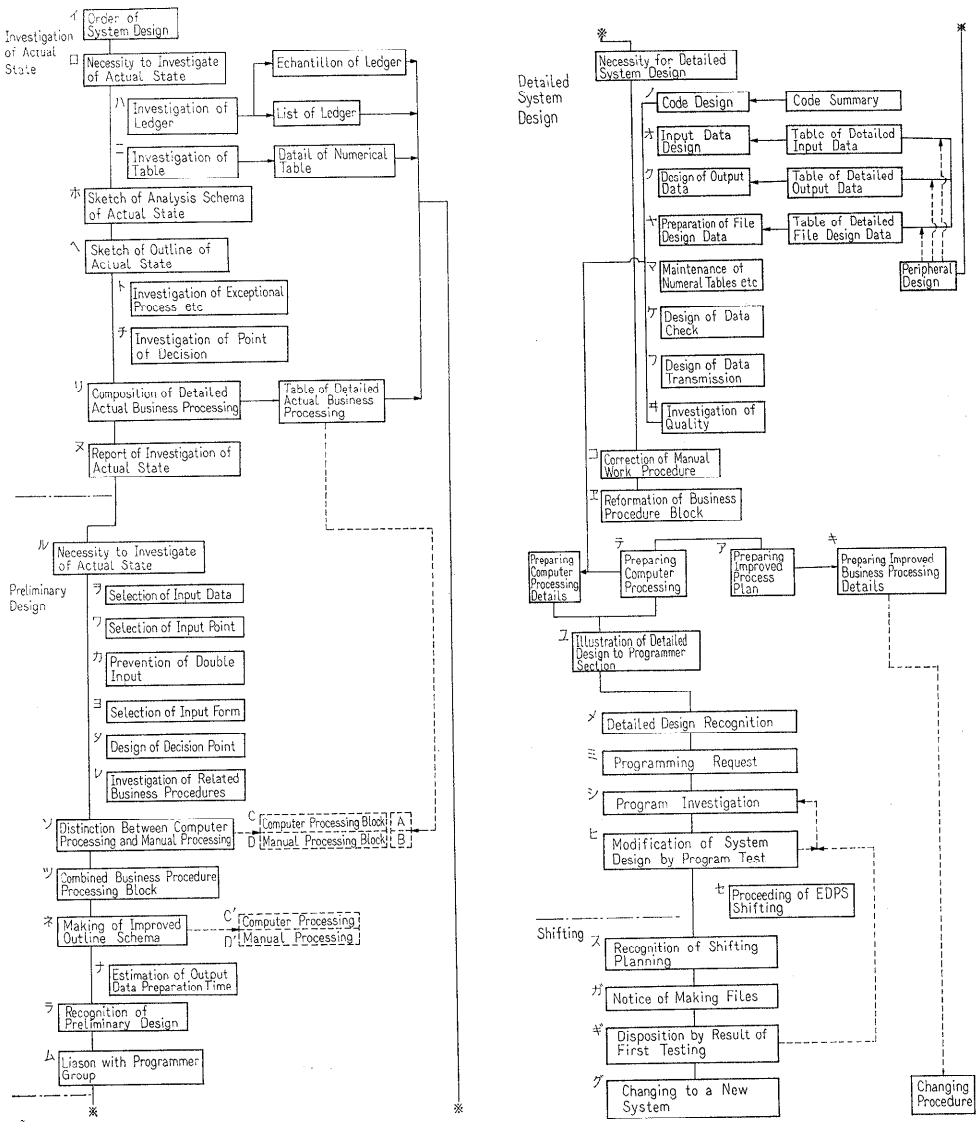

Fig. 2. Organization of EDPS system design.

Numerals in Fig. 3, are an example of Table 1. Estimated time is 128 electronic computer days and 72 manual days.

3. *Calculation and estimation of programming design time*

This estimation is accomplished by referring to the table. Points of the table are as following:

(1) This table is only used for COBOL Language programming.

(2) 'Risk factor' concept is introduced in constructing the tables.

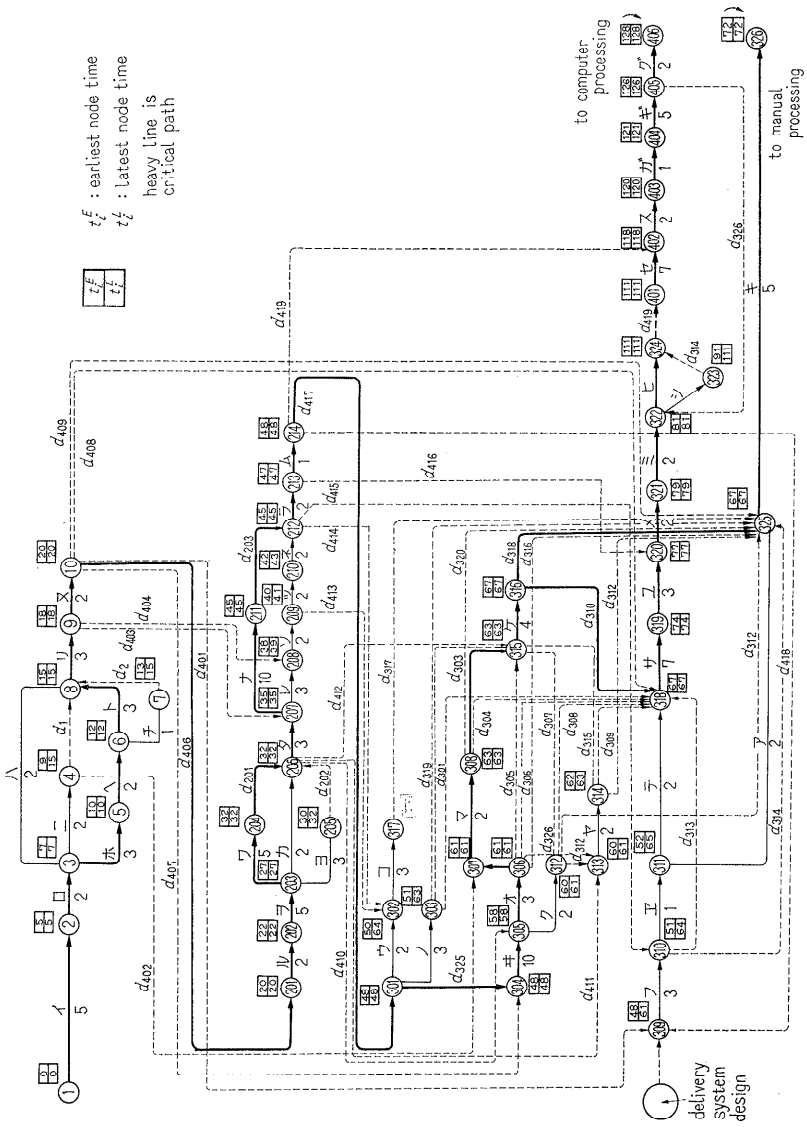

Fig. 3. PERT network system design of EDPS.

(3) Deciding relation of factors and levels, EDPS system design time is able to be estimated directly using data on the table. Factors are A composition of program, B number of input/output entries, C number of item and D processing difficulties. The 0 level is introduced in each factor, in order that the most suitable time can be estimated, even if sufficient information on 4 factors cannot be obtained. The relation of factor and level is shown in Table 2.

(4) When the programming technique progresses, even if programming becomes easier, in order that this method be adapted, the "processing difficulties" factor is introduced upon. Programming term and debbuging term in each factor and level which were calculated by actual data in past are shown in Table 3.

Table 1. Work List.

Node Number	Work Content	Symbol	Predecessor	Estimated day
(1, 2)	Order of System Design	イ	—	5
(2, 3)	Intention of Investigation of Actual State	ロ	イ	2
(3, 8)	Investigation of Ledger	ハ	ロ	2
(3, 4)	Investigation of Table	ニ	ロ	2
(3, 5)	Sketch of Analysis Schema of Actual State	ホ	ロ	3
(4, 8)	Dummy	d_1		0
(5, 6)	Sketch of Outline of Actual State	ヘ	ホ	2
(6, 8)	Investigation of Exceptional Proceeding etc.	ト	ヘ	3
(6, 7)	Investigation of Decision Pt.	チ	ヘ	1
(7, 8)	Dummy	d_2		0
(8, 9)	Composition of Detailed Actual Business Proceedure	リ	ハ, ニ ト, チ	3

From Table 3, Factor A has 5 levels, Factor B 6 levels, Factor C 4 levels and Factor D 6 levels. Therefore, 900 tables must be made for all the terms of A_i B_j C_k D_l (i, j, k and l is each level of A, B, C and D).

Then, for each character, a risk factor is designated.

Time distribution of programming and debugging is assumed to be Weibull distribution.

Probability density function is

$$f(x)=\frac{\beta}{\alpha}(x-\gamma)^{\beta-1}\cdot e^{-\frac{(x-\gamma)^{\beta}}{\alpha}} \quad (x\geqq\gamma) \qquad (1)$$

$$=0 \quad (x<\gamma)$$

Define 'risk factor', $g(x)=\int_t^{\infty} f(x)\cdot dx$

Then

$$g(t)=e^{-\frac{(t-\gamma)^{\beta}}{\alpha}} \qquad (2)$$

With 18 actual data of COBOL Language programming, (1) is the most approximate exponential distribution. Consequently, $\alpha=\mu_\alpha$, $\beta=1$, $\gamma=0$

$$g(t)=e^{-\frac{1}{\mu_\alpha}} \qquad (3)$$

The calculation and estimation method in this paper is made by substituting μ_α=(average value of construction time of A_i B_j C_k D_l) in (3) and calculating.

Table 2. Factors and Levels.

Factor	Illustration of Factor	Judgment of Level		Level
A Composition of Program	① Check, ② File Merge, ③ General Calculation, ④ Report Editing are selected as Program Function Elements. A Program has at least One Function Element. The Level used is Decided by Number of Function Elements that the Program must have. Error List is included in 'Report Editing'.	Number of Function Elements that Program must have	1	1
			2	2
			3	3
			4	4
B Number of I/O Entries	Level is decided by All the Number of I/O Entries that the Program Requires, but the Work File to compensate for the Lack of Internal Memory in the Same Program is omitted.	Number of I/O Program Files	2	2
			3	3
			4	4
			5	5
			Above 6	6
C No, of Items	Level is Decided by All the Number of Items Included in Above-Mentioned I/O File, but Relay I/O Entries for the Succeeding Program is omitted.		1~100	1
			101~300	2
			301~	3
D Difficulties in Process	Synthetic Supplement Factor of Above-Mentioned Factors EX., ① Programming Itself ② Skill of Programmer ③ Subroutine etc. ④ Other Factors	In Programming		
			Very Easy	1
			Easy	2
			Ordinary	3
			Slightly Difficult	4
			Considerably Difficult	5

Eeach Above-Mentioned Factor has a 0 Level and it is used in the Case of Unreliable Information.

Table 3. Programming term (week).

Level	Programming term				Debugging term		
	Facter *A*	Facter *B*	Facter *C*	Facter *D*	Facter *A*	Facter *B*	Facter *E*
1	3,409		4,907	2,887	1,648		3,131
2	5,246	4,581	5,058	3,489	4,345	4,053	3,519
3	8,920	4,927	5,968	4,694	7,719	4,123	4,004
4	12,594	5,619		5,899	11,093	4,120	4,489
5		6,311		7,104		4,296	4,975
6		7,003				4,383	
0	5,018	5,056	5,012	3,938	4,140	4,134	4,134

Tables obtained are 1) Correspondance of general 'risk' and individual 'risk' (Table 4.), 2) Calculation of individual program (Table 5.) and 3) Calculation of individual debug. Table 4 and Table 5 represent only a part of the required number of tables.

Table 4. Correspondenc of General 'Risk' and Individual 'Risk'.

General Risk Factor (%) \ Number of Activities	Number of Activities on Critical Path. Numeral in Table is Risk Fator (%) of an Individual Activity.																			
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Designation of Risk Factor (%) of Predetermined Scheduled Day 5	5	10	15	20	20	20	25	25	25	25	30	30	30	30	30	30	30	30	30	30
10	10	15	20	25	25	25	30	30	30	30	30	30	35	35	35	35	35	35	35	35
15	15	20	25	25	30	30	30	35	35	35	35	35	35	35	35	35	35	35	35	40
20	20	25	30	30	35	35	35	35	35	35	35	35	40	40	40	40	40	40	40	40
25	25	30	30	35	35	35	35	40	40	40	40	40	40	40	40	40	40	40	40	40
30	30	35	35	35	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40
35	35	35	40	40	40	40	40	40	40	40	45	45	45	45	45	45	45	45	45	45

Table 5. Calculation of Individual Program (week).

A	*B*	*C*	*D**	Risk Factor (%) of an Individual Activity													
				5	10	15	20	25	30	35	40	45	50	55	60	65	70
2	3	1	1	8.5	6.5	5.5	4.5	4	3.5	3	2.5	2.5	2	1.5	1.5	1.5	1
			2	10.5	8	6.5	5.5	5	4	3.5	3	3	2.5	2	2	1.5	1.5
			3	14	11	9	7.5	6.5	5.5	5	4.5	4	3.5	3	2.5	2	1.5
			4	18	13.5	11	9.5	8	7	6	5.5	4.5	4	3.5	3	2.5	2
			5	21.5	16.5	12.5	11.5	10	8.5	7.5	6.5	5.5	5	4.5	3.5	3	2.5
			0	15	11.5	9.5	8	7	6	5	4.5	4	3.5	3	2.5	2	2
		2	1	9	7	6	5	4.5	3.5	3	3	2.5	2	2	1.5	1.5	1
			2	11	8.5	7	6	5	4.5	4	3.5	3	2.5	2	2	1.5	1.5

* A : Composition of Program, B : Number of I/O Entries, C : Number of Items, D : Difficulties in Process.

Now, when a job is given, the procedure for its programming time estimation is as follows. First, general 'risk' is decided. If general 'risk' is assumed as 20% and the number of activities is 7, individual 'risk' is 35% according to Table 4. Next, each level in make up program A, number of I/O entries B, number of item C and difficulties of process D of its program is selected. If it is A_2 B_3 C_1 D_0, the result is 5 weeks according to Table 5. Debugging time is obtained in the same way.

4. *The most suitable time calculation for amendment and re-trial*

When the amended units reach r, they are sent from testing group to the amendment group which is composed of system design group and programming group. Average waiting time from an amended unit appearing to receiving modification is

$$W=\frac{1}{\mu}\left\{\sum_{j=1}^{r}\frac{\gamma_j}{1-\gamma_j}\right\}+\frac{r-1}{2}\left(\frac{1}{\mu}+\frac{1}{\lambda}\right) \quad [3]$$

Relation between W and $\rho(=\lambda/\mu)$ is shown in Fig. 4.

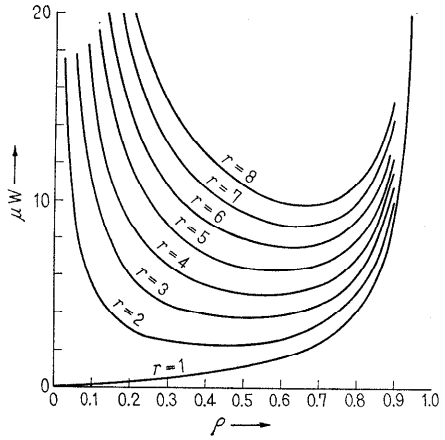

Fig. 4. Feedback indispensable times (r fixed system).

Now, amended group $=A$, testing group $=B$. N amended units are assumed to occur. Then, the amended units are numbered 1, 2, 3, ···, n and it is assumed that required amendment days are A_1, A_2, ···, A_i, ···, A_n and required retesting days are B_1, B_2, ···, B_i, ···, B_n. A Gantt chart for A and B work is shown in Fig. 5. While unit 1 is amended in A_1 days, B is idle. After A_1 days, B works.

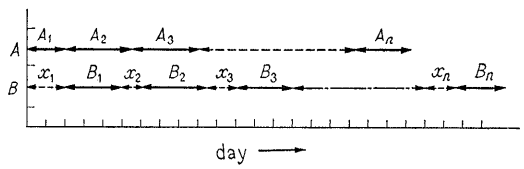

Fig. 5. Gantt chart for A and B work.

A is always busy, because retesting is necessarily preceded by an amendment.

Consequently, total proccssing time until re-trial is completed is calculated as time elapsed from the start of A_1 to the end of B_n.

If x_i represents the idle days just before B_i, the total testing time is $\sum_{i=1}^{n} B_i$, total idle time is $\sum_{i=1}^{n} x_i$ and total elapsed time is

$$T=\sum_{i=1}^{n-1}(B_i+x_i).$$

Put,

$$X(n)=\sum_{i=1}^{n} x_i$$

Then,

$$X(x)=\max\left[\left(\sum_{i=1}^{n} A_i-\sum_{i=1}^{n=1} B_i\right),\ \left(\sum_{i=1}^{n-1} A_i-\sum_{i=1}^{n-2} B_i\right),\ \cdots\cdots\right.$$
$$\left.\cdots\cdots,\ \left(\sum_{i=1}^{2} A_i-\sum_{i=1}^{1} B_i\right),\ \sum_{i=1}^{1} A_i\right]$$

$$= \max_{1 \leq r \leq n} \left(\sum_{i=1}^{r} A_i - \sum_{i=1}^{r-1} B_i \right)$$

This problem is to minimize the processing sequence of $X(n)$ [4], [5]. Consider one work processing sequence L (1, 2, ···, $m-1$, m, $m+1$, ···, n) and the other processing sequence L' (1, 2, ···, $m-1$, $m+1$, m, $m+2$, ···, n) that replaces m with $(m+1)$,

if $\min(A_{m+1}, B_m) \leqq \min(A_m, B_{m+1})$ (4)

then, $\max[F(m), F(m+1)] \geqq \max[F'(m), F'(m+1)]$ (5)

Consequently, from (4) an (5) the procedure to obtain the most suitable processing sequence is as follows.

Procedure 1

For all A_p, B_p look for minimum value $\min_p(A_p, B_p)$

Procedure 2

If it is A_s, replace s-th amended unit at the 1st position; if it is Bt, replace the t-th unit at the last position.

Procedure 3

Next, excepting s-th unit or t-th unit, repeat procedure 1 and procedure 2 for the rest unit and repeat them until the sequence of all the amended units is decided.

Now, Table 6 is obtained by investigating the amended unit ($r=5$) which are sent by r fixed system. Then the most suitable processing sequence 4→5→2→1→3 is obtained. Consequently, a new Gantt chart, as shown is Fig. 6, is obtained, from which it is determined that all the elapsed time is 31 days and the minimum idle days value is 2 days.

5. *Conclusion*

The caluculation and estimation of EDPS system design time must be easy

Table 6. Data ($r=5$) to be Corrected.

Section / Corect Unit	A_i	B_i
1	7	4
2	5	8
3	6	3
4	2	6
5	4	8

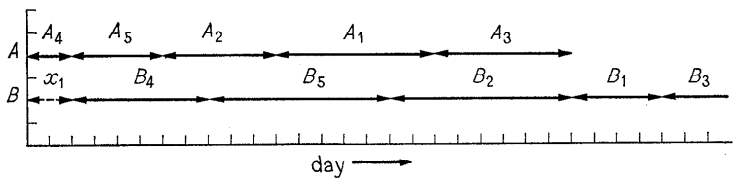

Fig. 6. Most suitable procedure series for Fig. 5.

and exact. In this paper, the calculation and estimation of EDPS system design time can be solved by constructing PERT network, programming by tables, checking feedback by Graphs, amendment and re-trial by Gantt chart, on a reliable basis than usual.

References

[1] System study group in NTT: Introduction of system design, Japan Business Efficiency Association (1964).
[2] Awano, T.: Advantages of New Linear Type Management Information System Design, *Keiei-Kagaku*, 13-2 (1970).
[3] Awano, T.: Devising Ways and Means to Calculate Indispensable Time in Management Information System Design, *Keiei-Kagaku*. 14-2 (1970).
[4] Johnson, S. M.: Optimal two-and three-stage production schedules with set up times included, *Nav. Res. Log. Quart.* (1-1954).
[5] Bellman, R.: Mathematical aspects of scheduling theory, *RAND Report*, 651 (1955).