

プログラム作成に対し影響の度合いを与える 要因の抽出について*

栗野敏雄*

Abstract

A programming time calculation method has been reported, wherein elements which influence programming and debugging time were selected¹⁾. This paper describes the method of selecting these elements.

First answers for the enquête were obtained from 66 programmers (Table 2.1) and a correlation matrix made by product moment method from them (Table 2.2). Then 4 factors were extracted by using the Centroid Method from the correlation matrix (Table 2.3). Next, to facilitate interpretation of the 4 factors, orthogonal rotation was made 6 times and loading factors were obtained (Table 2.4).

At last, selection of major factors H (electronic computer system), S (objective system), P (programmer) and M (management) was made by interpretation of Table 2.4.

Next, 4 clusters of each 4 factors were constructed. From them 4 elements were selected as representative of each 4 clusters. These were considered comparably objectively measurable elements and largely influential elements. That is, the number of input/output files from the H Factor, degree of complexity of program construction from the S Factor, data items from the P Factor and degree of processing difficulty from the M Factor. Next, each interaction of 4 elements was examined, and was recognized as negligible (Table 3.1).

These Factors were adopted as those influencing programming time. (including debugging time)

1. ま え が き

プログラム作成日程を推定する計算の一つの方法として、プログラマーがプログラム作成期間およびデバッグ期間に影響を与えると考える若干の要因を用いて、作成日程の推定を行なう方法が発表されている¹⁾。

プログラム作成期間やデバッグ期間の分布法則は、Weibull 分布と考えるとよいから、その確率密度関数は、

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

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

ここに、

α : 尺度を決めるパラメータ

β : 分布の形を決めるパラメータ

γ : 分布の位置を決めるパラメータ

で表わすことができる。

危険率を、

$$g(x) = \int_x^{\infty} f(x) dx \quad (2)$$

で定義すると、(1)より、

$$g(x) = e^{-\frac{(x-\gamma)^\beta}{\alpha}}$$

を得る。ここに、危険率とはあるジョブのプログラム完了予定日が決められたものと仮定したとき実行の結果、その予定日を越えるかもしれない確率のことである。

$\alpha = \mu_\alpha, \beta = 1, \gamma = 0$ とすると、

(1)より、

$$f_1(x) = \frac{1}{\mu_\alpha} \cdot e^{-\frac{x}{\mu_\alpha}} \quad (x \geq 0)$$

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

* Extraction of influential programming factors, by Toshio Awano (Nippon Telegraph and Telephone Public Corporation)

** 日本電信電話公社

したがって、(2)より危険率 $q_1(t)$ は、

$$q_1(t) = \int_0^{\infty} f_1(x) \cdot dt = e^{-\frac{t}{\mu_a}}$$

となり、 $q_1(t)$ は μ_a により決定することができる。前出の発表論文においては μ_a の要因として、A. プログラムの構造、B. I/O 数、C. 項目数、D. 難易性が用いられている。本論文においては、これらの要因の抽出について述べられている。

2. プログラム作成に関し影響の度合いを与える因子の分析

本調査に先だち重要なことは、問題領域として設定された分野を代表できる、完全な質問項目を作成することである。しかし、この方面における問題領域の研究が十分進んでいないため、この点については十分な考慮をほらい、次項に示すような 21 項目を選定して実験が行なわれた。

2.1 調査方法

調査は、まず、電々公社中央統計所のプログラマー全員 80 名について、表 2.1 のごとき調査表の 21 項目につき評定尺度の形式により、5 段階に分類して評定してもらった。調査は無記名方式としたのであるが、経験年数は記入してもらった。回答された調査表のうち 14 枚は記入もれや記入まちがいが発見されたので、それを除き 66 枚について分析計算を行なった。

分類は表 2.1 に示されるごとく、各項目ごとに a, b, c, d, e [a は最も関連度合いのあるもの、b はかなりある、c は中間(普通)である、d はほとんどない、e は最もない] の 5 段階とし、各欄のうち、適当と思われるものをプログラマーが現在有している常識から判断して、一つだけ選ぶようにした。項目の選定については先に述べたとおりであるが、プログラミングの困難性に関連のある要因を経験上から取り上げて、

下記のとおりとした。

- (1) データの項目数
データ処理加工の対象となる項目の入出力延項目数
- (2) E. D. P. システムの構成
単一 CPU, 主従 CPU, 共通ファイル複数 CPU などをいう
- (3) 入出力ファイル数
プログラムが必要とするすべての入出力ファイル数(ワークファイルを除く)
- (4) 汎用プログラムの開発状況
いかほど汎用プログラムが開発され、有効に使用されうる状況にあるかをいう
- (5) プログラミング技法の標準化の度合い
プログラミング技法の諸標準の開発状況がプログラム作成に影響する度合いをいう
- (6) 適用プログラム言語
FORTRAN, COBOL, アセンブラなどの使用言語による影響
- (7) 電算機(周辺装置を含む)の設備条件
電子計算機の記憶容量, 周辺装置の種類およびその設備数など
- (8) センターにおける機械稼働の余裕度
電子計算機の稼働余裕度がプログラム・デバッグのための使用時間のつごうに及ぼす影響をいう
- (9) 関連ジョブの有無
対象業務に関連するジョブの有無をいう
- (10) データ処理方式
バッチ処理, リアルタイム処理などのような処理の方式をいう
- (11) システム・プログラムの機能レベル
オペレーティング・システム・プログラムの有する機能のプログラム作成に与える影響の度合い

表 2.1 プログラム作成のため関連性のある因子分析調査表とその頻数分布
(Enquête table for factor analysis in programming and its summary) 経験年数 年 月

関連度合いの段階	項目	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)	(13)	(14)	(15)	(16)	(17)	(18)	(19)	(20)	(21)
		データの項目数	E・D・P システムの構成	入出力ファイル数	汎用プログラムの開発状況	技法の標準化の度	適用言語	電算機(周辺装置を含む)の設備条件	センターにおける機械稼働の余裕度	関連ジョブの有無	データ処理方式	システムの機能レベル	適用オペレーティングシステム	電算機の使用経緯	ジョブの複雑性	プログラムの量	稼働要員数	対象業務の知識	プログラムの複雑度	プログラマーの熟練度	プログラマーの経験	プログラムの難易
a		2	2	0	3	4	4	3	3	2	1	5	5	2	1	3	1	18	10	13	6	8
b		4	8	11	9	8	7	14	14	13	12	12	12	17	17	16	24	12	18	24	25	29
c		18	21	20	20	24	33	20	22	28	33	26	28	28	32	30	20	21	29	20	28	22
d		16	28	26	22	18	13	27	15	19	18	19	13	17	15	14	13	11	5	5	7	7
e		26	7	9	12	12	9	2	12	4	2	4	8	2	1	3	8	4	4	4	0	0

(12) 適用オペレーティング・システム

単一ジョブの処理, 多重ジョブ処理, 時分割処理など処理をするオペレーティング・システムの種類の影響の度合い

(13) 電算機の使用経験の度合い

実際にその電子計算機がどれほど長く使用されているかということが, プログラミングに影響する度合いをいう

(14) ジョブネットワークの複雑性

ジョブのシステム設計についてのネットワークの複雑の度合いをいう

(15) プログラム量

プログラム処理ステップ数, パスの回数をいう

(16) プログラマーの稼動要員数

与えられた対象業務に対して, 稼動できるプログラマーの要員をいう

(17) プログラマーの対象業務の知識の有無

プログラマーが作成せんとするプログラムの対象業務についての知識・経験などの有無をいう

(18) プログラムの機能構造の複合度

作成せんとするプログラムについて, チェック, ファイル突き合わせ, 計算, 一般処理, リポート編集などの機能要素の複合度をいう

(19) プログラマーの熟練度

プログラマーのプログラム作成についての年数・経験・訓練など, 一般的にみた熟練の度合い

をいう

(20) プログラミング期間

与えられたプログラミング期間の長短の及ぼす影響

(21) プロセスの難易性

事務処理・科学技術・経営科学・経営管理・情報検索・自動制御, またはシステム・プログラムなど, プログラムそのもののプロセスの難易性とサブルーチンの整備状態や担当プログラマーがそのプログラムに習熟していないことから生ずる難易性, そのほか各種の不均衡のために生ずる難易性も含んでいる

調査にあたっては, a, b, c, d, e をそれぞれ, 2, 5, 8, 5, 2 というような割合になるよう指示を与えた. 表 2.1 に, これら 21 項目の評定に対する頻数分布が示されている. この頻数分布表より, 偏差積法を用いて相関係数を求めた. その相関表が表 2.2 である.

2.2 セントロイド法による因子分析

表 2.2 の相関係数をセントロイド法により計算して因子の抽出を行なった²⁾. 得られた因子行列は, 表 2.3 に示されている.

相関行列において, 2変数 j, k 間の相関を m 個の共通因子で表わせば,

$$r_{jk} = a_{j1}a_{k1} + a_{j2}a_{k2} + \dots + a_{jm}a_{km} \quad (3)$$

相関行列における相関の和を求めると,

$$\sum_k \sum_j r_{jk} = \sum_m (\sum_j a_{jm})^2 \quad (4)$$

表 2.2 相関係数表 (Correlation matrix)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
1																					
2	.372																				
3	.414	.986																			
4	.570	.970	.970																		
5	.576	.854	.848	.930																	
6	.394	.628	.608	.710	.914																
7	.078	.944	.928	.836	.710	.538															
8	.474	.736	.812	.840	.972	.858	.702														
9	.202	.832	.832	.818	.882	.878	.858	.918													
10	.198	.774	.766	.776	.880	.920	.790	.906	.990												
11	.158	.846	.822	.960	.870	.866	.878	.864	.990	.984											
12	.274	.632	.640	.852	.872	.976	.626	.914	.932	.964	.912										
13	.058	.434	.746	.698	.772	.812	.824	.888	.974	.966	.960	.914									
14	.070	.658	.678	.654	.772	.856	.752	.894	.966	.970	.948	.946	.992								
15	.082	.634	.654	.642	.778	.876	.720	.892	.958	.970	.942	.960	.992	.994							
16	.008	.410	.548	.448	.460	.476	.590	.804	.710	.676	.642	.652	.812	.794	.782						
17	-.074	.126	.492	.004	.226	.526	.270	.146	.480	.550	.550	.542	.520	.472	.464	.120					
18	-.160	.158	.176	.180	.446	.736	.292	.614	.684	.748	.668	.820	.772	.840	.858	.640	.746				
19	-.530	-.168	-.114	-.208	.028	.272	.126	.284	.364	.402	.344	.444	.546	.588	.592	.656	.612	.844			
20	-.262	.234	.288	.218	.190	.586	.460	.652	.712	.740	.688	.756	.982	.870	.872	.838	.618	.934	.910		
21	-.442	.072	.146	.020	.126	.336	.368	.466	.532	.538	.488	.532	.702	.716	.704	.808	.514	.846	.956	.994	

表 2.3 得られた因子行列 (Factor matrix obtained)

	A ₁	A ₂	A ₃	A ₄	h ² obt.	h ² guess	差
1	.18	.63	.32	.29	.61	.58	.03
2	.73	.63	-.28	-.24	1.04	.99	.05
3	.79	.48	-.27	-.20	.97	.99	-.02
4	.77	.69	-.20	.17	1.12	.97	.15
5	.84	.54	.24	.09	1.05	.97	.08
6	.88	.15	.41	.03	.96	.98	-.02
7	.79	.27	-.49	-.18	.96	.95	.01
8	.93	.23	.09	.38	1.05	.97	.08
9	.99	.12	-.06	-.06	.99	.99	.00
10	.99	.07	.07	-.07	.98	.99	-.01
11	.98	.15	-.11	-.13	1.00	.99	.01
12	.97	.03	.26	.13	1.01	.98	.03
13	.98	-.18	-.12	.18	1.03	.99	.04
14	.98	-.14	-.03	.05	.98	.99	-.01
15	.98	-.17	.05	.07	.99	.99	.00
16	.76	-.28	-.29	.34	.84	.84	.00
17	.52	-.30	.23	-.59	.75	.75	.00
18	.77	-.57	.36	-.07	1.03	.93	.10
19	.47	-.86	.06	-.06	.96	.96	.00
20	.79	-.67	-.12	.10	1.08	.99	.09
21	.62	-.75	-.18	.04	.97	.99	-.02

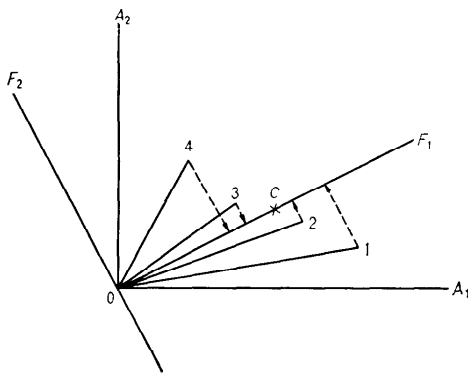


図 2.1 セントロイド法の原理 (Centroid method principle)

m 個の共通因子をもつ n 個のテストの場合には、その重心の座標は、

$$\frac{1}{n} \sum_j a_{j1}, \frac{1}{n} \sum_j a_{j2}, \dots, \frac{1}{n} \sum_j a_{jm} \quad (5)$$

となる。

図 2.1 に示すように、この重心を通るようにして第 1 軸 F_1 を引き、この F_1 に直交する軸を F_2, F_3, \dots, F_m とすれば、この重心は F_2 から F_m までの $m-1$ 個の軸に関しては、その投影は 0 になる。すなわち、これらの新しい軸への各テストの投影を $a'_{j1}, a'_{j2}, \dots, a'_{jm}$ とすれば、

$$\sum_j a'_{j2} = \sum_j a'_{j3} = \dots = \sum_j a'_{jm} = 0 \quad (6)$$

したがって、重心 C の座標は、

$$\frac{1}{n} \sum_j a'_{j1}, 0, 0, \dots, 0 \quad (7)$$

となる。

(6) を (4) に代入すると、

$$r_i = \sum_k \sum_j r_{jk} = \left(\sum_j a'_{j1} \right)^2 \quad (8)$$

(8) を第 1 因子負荷量 a'_{j1} について解くと、

$$\sum_j a'_{j1} = \sqrt{r_i} \quad (9)$$

(3) より、

$$\sum_j r_{jk} = a'_{k1} \sum_j a'_{j1} + a'_{k2} \sum_j a'_{j2} + \dots + a'_{km} \sum_j a'_{jm} \quad (10)$$

(6) を (10) に代入すると、

$$\sum_j r_{jk} = a'_{k1} \sum_j a'_{j1} \quad (11)$$

(11) に (9) を代入し、 a'_{k1} について解くと、

$$a'_{k1} = \frac{\sum_j r_{jk}}{\sqrt{r_i}} \quad (12)$$

(12) から各項目の F_1 軸に対する因子負荷量が求められ、これより第 1 因子が求められる。

第 1 因子負荷量が求められたら、

$$1r_{jk} = r_{jk} - a'_{j1} a'_{k1}$$

により、第 1 因子残差 $1r_{jk}$ を算出し、これに基づいて第 2 因子負荷量を求める。同様の方法により、第 3 因子以下が抽出される。

因子を抽出するごとに残差は小さくなるから、もうこれ以上の因子を抽出しても意味がないと判断された場合に、因子の抽出を止めることとする。

次に、単純構造 (simple structure) を見いだして因子の解釈を比較的容易にするために、得られた因子負荷量の軸回転を行なう (簡単のため直交回転を用いた)。

$A = (a_{ij})$ を初期の因子行列、 $F = (f_{ij})$ を最終の因子負荷行列、 $T = (t_{ii'})$ を直交変換行列とすると、

$$F = A \cdot T$$

いま、因子が 2 個の場合について考える。このとき観測特性は 2 本の参考軸による平面上の点で座標が定められる。そして、この軸をどのように回転しても、共通因子空間で座標が変わるだけであって、その相関の大きさに変化はない。いま、図 2.2 において、初期の回転前の軸を A_1, A_2 、回転後の軸を F_1, F_2 で、各点 $P_j (j=1, 2, \dots, p)$ を P で表わして、回転の関係式を導く。

点 P の座標を回転前と回転後とで、それぞれ (a_1, a_2) および (f_1, f_2) とすると図 2.2 より、回転による

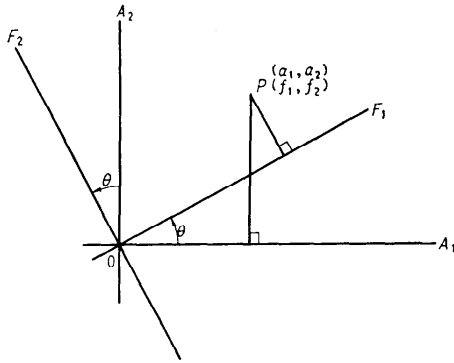


図 2.2 平面上の軸の回転 (Orthogonal rotation)

各座標間の関係は、

$$f_1 = a_1 \cos \theta + a_2 \sin \theta$$

$$f_2 = -a_1 \sin \theta + a_2 \cos \theta$$

となり新因子負荷量を得る。

各点 j については、

$$F = \begin{pmatrix} f_{11} & f_{12} \\ f_{21} & f_{22} \\ \vdots & \vdots \\ f_{p1} & f_{p2} \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ \vdots & \vdots \\ a_{p1} & a_{p2} \end{pmatrix} \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix}$$

$$\equiv A \cdot T$$

A が A_1, A_2, A_3, A_4 の 4 軸の場合 F_1, F_2, F_3, F_4 を求める手順は次のようである。

	軸の組	回転角	新軸	固定している直交軸
第 1 回目	A_1, A_2	$\theta_{1,2}$	B_1, B_2	A_3, A_4
第 2 回目	B_1, A_3	$\theta_{1,3}$	C_1, B_3	B_2, A_4
第 3 回目	C_1, A_4	$\theta_{1,4}$	F_1, C_4	B_2, B_3
第 4 回目	B_2, B_3	$\theta_{2,3}$	C_2, C_3	F_1, B_4
第 5 回目	C_2, B_4	$\theta_{2,4}$	F_2, C_4	F_1, B_3
第 6 回目	C_3, C_4	$\theta_{3,4}$	F_3, F_4	F_1, F_2

上記の手順により、軸の直交回転を行なって得た因子負荷量の行列が表 2.4 である。これらの因子負荷量により因子の解釈を行なうこととする。

2.3 因子の解釈

表 2.4 の因子行列について解釈を行なうと、 F_1 因子(第 1 因子)について高い因子負荷量をもつ項目は、2. EDP システム構成, 3. 入出力ファイル数, 4. 汎用プログラム開発状況, 7. 電子計算機の設備条件, 8. センターにおける機械稼働の余裕度, 11. システム・プログラムの機能レベル, 13. 電子計算機の使用経験の度合いである。これらはすべて電子計算機とメーカー・サイドに関係の深いものである。したがって、これらを電子計算機自体により影響される因子と

表 2.4 軸回転後の因子行列 (Final factor matrix)

	F_1	F_2	F_3	F_4	h^2 obt	回転前の h^2	差
1	.21	-.09	.64	-.39	.61	.58	.03
2	.81	.43	.08	-.47	1.06	.99	.07
3	.80	.48	.08	-.31	.97	.99	-.02
4	.90	.22	.47	-.27	1.15	.97	.18
5	.59	.48	.64	-.28	1.06	.97	.09
6	.34	.68	.61	.00	.95	.98	-.03
7	.87	.45	-.11	-.09	.97	.95	.02
8	.69	.39	.64	.16	1.06	.97	.09
9	.68	.68	.26	-.07	.99	.99	.00
10	.58	.74	.32	.09	.99	.99	.00
11	.71	.69	.19	.01	1.01	.99	.02
12	.48	.67	.56	-.18	1.02	.98	.04
13	.67	.58	.27	.45	1.05	.99	.06
14	.59	.67	.27	.33	.98	.99	-.01
15	.52	.69	.33	.36	.98	.99	-.01
16	.64	.30	.16	.58	.86	.84	.02
17	-.08	.86	-.12	.04	.76	.75	.01
18	.12	.83	.27	.55	1.07	.93	.14
19	-.02	.59	-.11	.78	.96	.96	.00
20	.34	.60	.02	.69	.95	.99	-.04
21	.27	.53	-.13	.79	.99	.99	.00

して取り上げ、 H 因子とする。

F_2 因子 (第 2 因子) について高い因子負荷量をもつ項目は、9. 関連ジョブの有無, 10. データ処理方式, 14. ジョブネットワークの複雑性, 15. プログラム量, 17. プログラム機能構造の複合度である。これらは、すべて対象業務のシステムにより影響される要因である。これらを一つの因子として取り上げ S 因子とする。

F_3 因子 (第 3 因子) について高い因子負荷量をもつ項目は、1. データの項目数, 5. プログラミング技法の標準化の度合い, 6. 適用プログラミング言語, 12. 適用オペレーティング・システムである。これらは、すべてプログラマー自体で決定したり、開発したりせねばならないことが多い問題である。したがってこれらをプログラマーにより影響される因子として取り上げ、 P 因子とする。

最後に、 F_4 因子 (第 4 因子) について、高い因子負荷量をもつ項目は、16. プログラマーの稼働要員数, 19. プログラマーの熟練度, 20. プログラミング期間, 21. プロセスの難易性である。これらは、管理者により影響される要因であるから、これらを一つの因子として取り上げ M 因子とする。かくして、おのおの独立した H, S, P, M の 4 因子を決定する。これらの因子群を図 2.3 に示す。

3. 要因の決定

因子分析の結果、プログラム作成のために困難の影

- H** 因子 (電子計算機自体を軸とするもの)
- EDP システム構成
 - 入出力ファイル数
 - 汎用プログラム開発状況
 - 電子計算機設備条件
 - センターにおける機械稼働の余裕度
 - システムプログラムの機能レベル
 - 電子計算機の使用経験の度合い
 - 記憶容量
 - I/O チャンネルの数
 - 使用開発電子計算機
 - 特殊表示用機器
 - 使用ランダムアクセス装置
 - ビット数/1 語
 - メモリ・アクセス時間
 - プログラミング用電子計算機と実際に使用する電子計算機とが同じか違うか
 - 設備運用の型
- S** 因子 (対象業務のシステムを軸とするもの)
- 関連ジョブの有無
 - データ処理方式
 - ジョブネットワークの複雑性
 - プログラム量 (入力量・出力量)
 - プログラム機能構造の複合度
 - 設計条件の定義の不明確
 - 作成プログラムに対する独創性の要求
 - システム設計書の良否
 - 組織上の利用者の数
 - 応答時間の条件
 - 設計の安定性
 - オン・ラインの条件
 - 入力様式数
 - 出力様式数
 - 他のシステムとのインターフェイスの複雑性
 - 事務命令率
 - 入出力率
 - 論理判断率
 - 自動チェック率
 - 情報蓄積検索機能率
 - データ入出力率
 - 制御規制率
 - 利用者の未経験
 - 他のシステムとのインターフェイスの複雑性
 - システムの情報源の数
 - 開発システムの変更予定の程度
 - システムの機能数
 - システムのコンポーネントの数
 - 老朽化しないシステムコンポーネントの数
 - 意思決定機能率
 - タイミングの逼迫さの条件
 - 書式総数
 - プログラムの型 (事務用, 科学用, ユティリティ, 単独処理, その他)
 - 最初のプログラムか否か
 - 平均ターンアラウンド時間
- P** 因子 (プログラマー自体を軸とするもの)
- データ項目数
 - 入力の変数項の数
 - 出力の変数項の数
 - プログラミング技法の標準化の度合い
 - 適用プログラミング言語
 - 適用オペレーティングシステム
 - 運用条件の知識の不足
 - プログラム・インターフェイスの複雑性
 - 引渡しオブジェクト・プログラム命令総数
 - 引渡しオブジェクト・プログラム再使用命令数
 - 作成オブジェクト・プログラム非引渡し命令総数
 - 作成ソース・プログラム命令総数
 - オブジェクト・プログラム廃棄率
 - ソース・プログラム廃棄率
 - ブランチ条件数
 - データ・ベースの語数
 - データ・ベース項目種類数
 - 変換機能率
 - ジェネレーション機能率
 - サブ・プログラムの数
 - POL 拡張率
 - 支援プログラムの利用度
 - プログラミング・ステップで作成する書式
 - プログラミング・ステップからの書式で利用できる範囲
 - 使用アセンブラまたはコンパイラ
- M** 因子 (管理を軸とするもの)
- プログラマーの稼働要員数
 - プログラマーの熟練度
 - プログラミング期間
 - プロセスの難易性
 - 上級・中級・下級プログラマー率
 - プログラマーの平均言語経験
 - プログラマーの平均適用業務経験
 - プログラマーの設計参加率
 - プログラマーの勤続年数
 - 最大割り当てプログラマー数
 - 管理手続きの欠除
 - 共同設計部門数
 - プログラマー以外の電子計算機操作
 - オペレーション設計以外に關与するプログラマー
 - 人間の誤り
 - システム情報の入手しやすさ
 - 上級・中級・下級アナリスト率
 - 特殊用具の利用度
 - 方針, 手続きの標準化の度合い
 - 書類の検討度合い
 - プログラマーなど要員の回転
 - プログラマーの勤務条件
 - プログラム作成チームの編成要員
 - プログラマーの健康条件
- 情報源の書類の品質**
- テーブルの語数およびデータ・ベースにない常数項の数

図 2.3 プログラム作成に關し影響の度合いを与える因子の草むら
(Clusters of four influential programming factors)

響を与える因子は、4 因子で説明づけられることがわかった。そして、この 4 因子を軸として、図 2.3 のような四つの草むら (cluster) を得ることができた。草むらの各要因のうちから、比較的客観的に測定しやすい、しかも幅の広い影響力を有すると思われる要因を

代表として選出すると、第 1 次草むらからは入出力ファイル数、第 2 次草むらからはプログラム機能構造の複合度、第 3 次草むらからはデータの項目数、第 4 次草むらからはプロセスの難易性を得た。

以上のようにして、 μ_a を決定する要因 A, B, C, D

表 3.1 プログラム作成期間分散分析表 (Analysis of variance of programming time)

データの割付け (L_{16} 直交表)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	y
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1.62
2	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2	4.51
3	1	1	1	2	2	2	2	1	1	1	1	2	2	2	2	2.66
4	1	1	1	2	2	2	2	2	2	2	2	1	1	1	1	4.42
5	1	2	2	1	1	2	2	1	1	2	2	1	1	2	2	4.34
6	1	2	2	1	1	2	2	2	2	1	1	2	2	1	1	2.57
7	1	2	2	2	2	1	1	1	1	2	2	2	2	1	1	2.72
8	1	2	2	2	2	1	1	2	2	1	1	1	1	2	2	4.43
9	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	2.57
10	2	1	2	1	2	1	2	2	1	2	1	2	1	2	1	9.02
11	2	1	2	2	1	2	1	1	2	1	2	2	1	2	1	5.00
12	2	1	2	2	1	2	1	2	1	2	1	1	2	1	2	5.14
13	2	2	1	1	2	2	1	1	2	2	1	1	2	2	1	6.11
14	2	2	1	1	2	2	1	2	1	1	2	2	1	1	2	7.49
15	2	2	1	2	1	1	2	1	2	2	1	2	1	1	2	5.30
16	2	2	1	2	1	1	2	2	1	1	2	1	2	2	1	12.36
割付け	A	B	A×B	C	A×C	B×C		D	A×D	B×D		C×D				

分散分析表 (I)

要因	S	ϕ	V	F_0
A	41.345	1	41.345	7.863*
B	6.734	1	6.734	1.281
C	0.903	1	0.903	0.172
D	24.059	1	24.059	4.576
A×B	4.709	1	4.709	0.896
A×C	0.126	1	0.126	0.024
A×D	6.812	1	6.812	1.296
B×C	1.440	1	1.440	0.274
B×D	0.511	1	0.511	0.097
C×D	0.185	1	0.185	0.035
e	26.290	5	5.258	
				$F_{5,1}^1(0.05)=6.61$

分散分析表 (II)

要因	S	ϕ	V	F_0
A	41.345	1	41.345	13.311**
B	6.734	1	6.734	2.168
D	24.059	1	24.059	7.746*
A×D	6.812	1	6.812	2.193
$e' \begin{pmatrix} e, B \times C \\ C, B \times D \\ A \times B, C \times D \\ A \times C, \end{pmatrix}$	34.164	11	3.106	
				$F_{11,1}^1(0.05)=4.84$
				$F_{11,1}^1(0.01)=9.65$

注) **: 危険率 1% で有意, *: 危険率 5% で有意

が一応決定されたのである。

次に、これらの要因 A, B, C, D が互いに交互作用を有するか否かを検定する必要がある。表 3.1 のプログラム作成期間分散分析表は、プログラム作成期間の実態調査 (補注参照) により得られた 16 個のプログラム作成期間 (y 週) により、分散分析を行なって検定を行なったものである。

その分散分析表 (I) によれば要因間の分散比 F_0 の値は $A \times B$: 0.896, $A \times C$: 0.024, $A \times D$: 1.296, $B \times C$: 0.274, $B \times D$: 0.097, $C \times D$: 0.035 となり、 $F_{5,1}^1(0.05)=6.61$ よりも非常に小さいので、その交互作用はいずれも無視することができる。念のため、不偏分散 V の値が誤差項 e の V よりも大きいものを見ると、それは、A, B, D, $A \times D$ である。そこで、それ以外の要因をすべて誤差項にプールして分散分析を行なうと表 3.1 の分散分析表 (II) を得る。この表より交互作用 $A \times D$ の F_0 の値は 2.193 となり、これ

は $F_{11,1}^1(0.05)=4.84$ より小さいため、これを無視することができる。

デバッグ期間に対しても同様にして行なわれ、その結果同じく A, B, C, D 間の交互作用は無視することができる。したがって A, B, C, D をプログラム作成のための要因として決定したのである。

次に各要因の水準についての日程を算出しなければならぬ。水準とは、一つの要因内の階級のことである。各要因の水準の数を表 3.2 に示すが、前記期間に対して短かく作用するものから長く作用する順に水準 1, 水準 2, ……とする。各水準に対する平均日程の算出は、実測によるデータが非常に多ければ、もちろん

表 3.2 要因の水準数 (Factor levels)

	要因 A	要因 B	要因 C	要因 D
水準の数	4	5	3	5
水準の範囲	1~4	2~6	1~3	1~5

表 3.3 各要因の水準の平均日程
(Average time for each factor level)

単位: 週 (week)

対象	水準	要因 A	要因 B	要因 C	要因 D	総平均値
プログラム作成期間	1	3.409	4.368	4.779	3.790	5.016
	2	6.624	5.665	5.245	6.243	
デバッグ期間	1	2.658	4.053	4.131	3.640	4.134
	2	5.610	4.215	4.136	4.628	

表 3.4 旧水準と新水準との関係
(Relation of old and new levels)

旧水準	新水準			
	要因 A	要因 B	要因 C	要因 D
1	1.500	2.190	(57.9)	2.250
2	2.375	4.063	(211.5)	4.286

注) 因子 C は直接項目数で算出したので () 内は、旧水準に対応する項目数で表わしている。

表 3.5 プログラム作成期間 (Programming time)

単位: 週 (week)

水準	プログラム作成期間				デバッグ期間		
	要因 A	要因 B	要因 C	要因 D	要因 A	要因 B	要因 D
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	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

問題はないのであるが、それほど多くないので、次のような方法によって求めた。すなわち、これには表 3.1 の直交配列表を利用する。各要因の水準の平均値を求めると表 3.3 のようになる。ここまでは、水準は各要因について 2 水準の層化で処理が行なわれてきたのであるが、実際には上述表 3.2 のように、もっと細分化した水準の数が要求されている。したがって、表 3.3 の各要因の水準を表 3.2 の水準に変換を行なう必要がある。よって、それぞれを旧水準・新水準とし、それらの関係を基礎データから算出すると表 3.4 のようになる。すなわち、表 3.4 に掲げた各新水準に対応する各要因水準の平均日程が、表 3.3 に示された数値群ということになる。たとえば、プログラム作成期間の要因 B でいえば、新水準 2.190 に対応する日程が 4.368 週であり、新水準 4.063 に対応する日程が 5.665 週ということになる。この関係を利用して新水準が 2, 3, ..., 6 ならば、それぞれ日程はどうなるかを、直線内挿および直線外挿によって求めると表 3.5 を得る。そして、これにより与えられた水準による数値表

表 3.6 要因と水準 (Factors and levels)

要因	要因の説明	水準の判定	水準	
構造 A	①チェック、②ファイル突き合わせ、③計算処理一般、④リポート編集という四つの機能をプログラムの機能要素と呼び、プログラムには少なくとも一つの機能要素を対応させる。要因 A の水準の決定は、いま対象としているプログラムに何個の機能要素をもたせたいかによって決まる。エラー・リストの編集は機能要素「リポート編集」を対応させる。	そのプログラムにもたせたい機能要素の数が	1 個の場合	1
			2 個の場合	2
			3 個の場合	3
			4 個の場合	4
I/O 数 B	プログラムが必要とするすべての入出力ファイル数によって要因 B の水準が決まる。ただし同一プログラム内において内部記憶の不足を補う意味で使用するワーク・ファイルはこの対象から除く。	プログラムの入出力ファイルの数が	2 個の場合	2
			3 個の場合	3
			4 個の場合	4
			5 個の場合	5
			6 個以上の場合	6
			1~100 個の場合	1
項目数 C	上記入出力ファイルに含まれるすべての項目の数によって要因 C の水準が決まる。ただし単に後続プログラムに中継させるためだけに入出力する項目は対象から除く。		101~300 個の場合	2
			301 個以上の場合	3
			1~100 個の場合	1
難易性 D	前記 3 要因 (要因 A, B, C) の補正要因として、以下の事項を総合的に補正する。 ①プログラムそのものの難易性 ②担当プログラマーの熟練性 ③サブルーチンの整備状態 ④その他各種の不均衡	プログラムの難易性の点で	かなり容易と考えられる場合	1
			やや容易と考えられる場合	2
			普通と考えられる場合	3
			やや困難と考えられる場合	4
			かなり困難と考えられる場合	5

上記各要因について水準 0 を設け、その要因の水準が不確定状態の場合に適用する。

を求めることができる。

かくて、表 3.6 に示すような要因 A, B, C, D と水準の表が与えられれば、この表を利用して前記の数値表により、プログラム作成日程を推定することができる。この表 3.5, 表 3.6 が既発表論文¹⁾の表 3, 表 2 である。

4. むすび

以上のようにして、セントロイド法によりプログラム作成に関し影響の度合いを与える 4 個の因子を抽出し、その因子を軸として、それぞれ作られた四つの草むらのなかから選出せられた 1 個ずつの要因、すなわち、A. プログラムの構造、B. I/O 数、C. 項目数、D. 難易性をプログラムに作成日程に関係する要因として決定したのである。四つの草むらをそれぞれ代表しうような要因であるならば、他の要因でもよいわけである。このような考え方により得られた要因が、従来行なわれていたステップ数を基礎とするプログラム作成日程推定よりもより信頼性のあるプログラム作成日程推定¹⁾に適用されており、特に、COBOL を使

