電力系統制御用ソフトウェアの階層化分割法について (その2)*

——階層分割の定量的評価とソフトウェアシステムの最適設計手法——

奈良宏一**
辻　俊彦***

(昭和53年4月28日受理)

Hierarchical Decomposition of Software System for Power Systems Computer Contral (Part 2)

— Ovantitative Evaluation of Hierarchical Decomposition and Designing Technique of Optimum Software System —

by Koichi NARA and Toshihiko TUJI

The software quality of the computers for the power system's control must satisfy some basic requirements; maintainability, modifiability, reliability, be understandable etc.

So, in the previous paper, we argued the necessity of hierarchical decomposition of the software system for the power system's control, and proposed a method of hierarchical software decomposition which makes it easy to document, to coodinate and to maintain the software system.

In this paper, we propose a software designing technique by which we can design optimum software system by evaluating the hierarchical software decomposition proposed in the previous paper.

Although the evaluating technique used in this paper is a popular one, as far as the coefficients of the evaluation functions are adequate to the system and programming team, the softwave system designed by this technique would be an optimum one.

The proposed technique can be used in other control systems in addition to the power system's control by ajusting the method and criterions of evaluation for them.

1. まえがき

電力系統制御用計算機においては，電力系統の拡張に伴うソフトウェアの増設・変更，広範囲な地域からの大量データの多目的処理，制御の高信頼性といった問題があり，ソフトウェ

* 電気学会情報処理研究会 (昭和52年12月) で一部発表
** 北見工業大学電気工学科
*** 三菱電機株式会社電力系統システム部

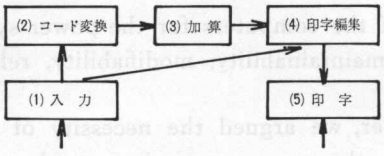
アの設計段階において，これらの要求を十分に満足した設計がなされなければならず，また，一般にソフトウェアを設計・保守する技術者は電力系統制御を専門とする技術者であり，プログラムの作成に十分な経験を積んでいないことから，理解し易いプログラム構造を有することが要求される。

そのため，前稿[1]では，電力系統制御用ソフトウェアにおける階層化分割の必要性とその目的について説明し，階層の概念とプログラムモデルの定性的な検討結果から，記述・統合・保守が容易で，高信頼性を実現可能なソフトウェアの階層化分割法について提案した。

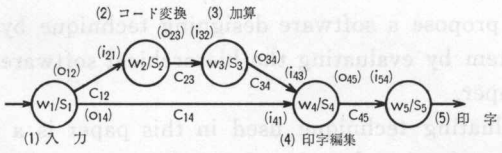
本稿では，前稿で提案した階層化分割法により分割されたシステムに，ソフトウェアに対する要求の観点から評価を加え，最適な階層分割を得るソフトウェアシステム設計手法について提案する[2]。

2. 階層分割の評価によるソフトウェアシステム最適設計手法

前稿で提案した分割手法において，ソフトウェアの分割規準があまり厳しすぎると，分割された要素が多くなりすぎ，製作人工が大きくなって分割結果が非現実的なものとなる。そのため，適当な厳しさの規準を設けてソフトウェアシステムを分割し，得られたシステムの要素

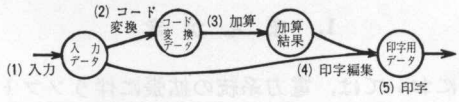


(a) 簡単なソフトウェアシステムの例
(a) An example of simple software system



(b) 例題の評価グラフ
(b) An evaluation graph of the example

注) w_i; i プログラムのワード数　S_i; i プログラムの評価に関するデータ
Oij, i_{ij}; プログラム ij 間で受け渡されるデータ数
Cij; i プログラムと j プログラムの結合について評価するためのデータ



(c) 例題のデータフロー図
(c) A data flow chart of the example

図—1 簡単なソフトウェアシステムの評価グラフとデータフロー図

Fig. 1. An evaluation graph and a data flow chart for a simple soft-ware system.

を順次消去または統合しながら，与えられた評価規準によって評価し，評価結果が最適となるシステムを採用するのがシステム設計手順からいっても現実的である。したがって，ここでは以下に示す手順を採っている。

まず，分割されたシステムを検討するため，ソフトウェアシステムをプログラムをノードプログラム間の従属関係（データの授受，起動・制御の関係等）を有向アークとした評価グラフに描く。たとえば，図-1 (a) に示す簡単なシステムを評価グラフの形に描き直すと，図-1 (b) のようになる。

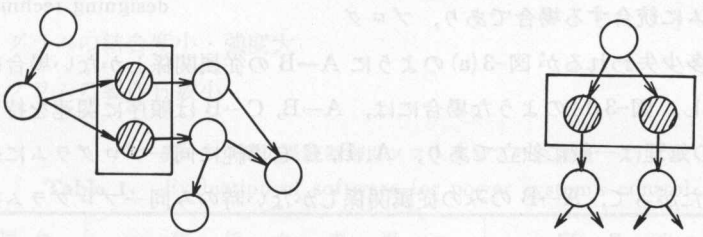
次に，評価グラフによって，プログラムの消去・統合の可能性を検討する。

提案した分割手法から，消去可能なプログラムは分割されたシステムの要素として生じないはずであるが，一般に，2種類以上のプログラムについて単純に消去または統合できるのは同一または連続する2つの処理が存在する場合であり，次の2通りのケースが考えられる。

(1)　2つのプログラムが全く同一業務，同一機能を有する場合（図-2参照）。

(2)　プログラムAとプログラムBにおいてA, B間のみの単純な従属関係しか存在しない場合（図-3 (a) 参照）。

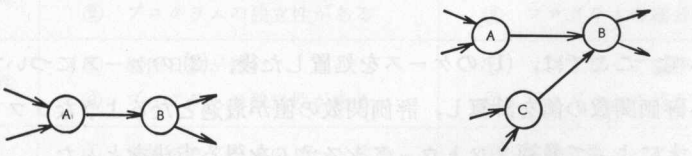
前稿で提案したソフトウェア分割手法を用いる限り，(1) のケースは発生し得ない。なぜなら，同一業務であれば業務分割によって2つ以上に分割され得ず，同一機能であれば，機能分割によって2つ以上に分割され得ないからである。しかしながら，仮に，分割時の誤り等によって (1) のケースが発生したとしても，以下の手順で簡単に余分なプログラムの消去が可能



(a)　同一業務が存在する場合　　(b)　同一機能が存在する場合
(a)　Existence of same works　　(b)　Existence of same functions

図—2　同一業務又は同一機能が存在する場合の評価グラフ

Fig. 2.　Evaluation graphs having same works or same functions.



(a)　単純な従属関係しかない場合　　(b)　複雑な関係がある場合
(a)　A case of simple relation　　(b)　A case of complicated relations

図—3　プログラム間の関係の例

Fig. 3.　An examples for relations of programs.

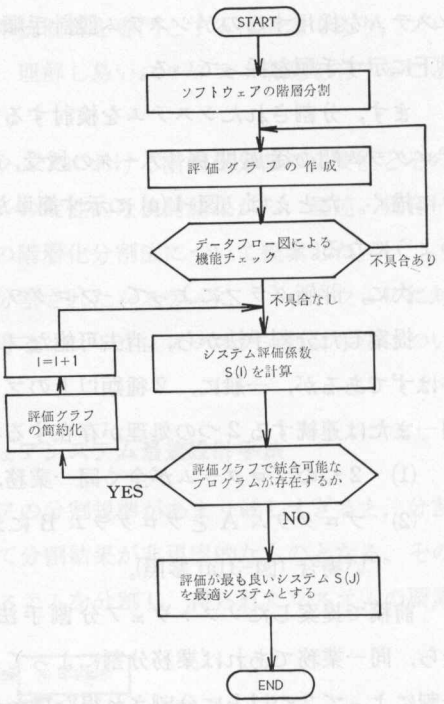
である。

(a) 評価グラフをデータをノード，プログラムを入力が矢，出力が根の有向アークとして描き直す。(このグラフをデータフロー図と呼び，図-1(b)に示す評価グラフのデータフロー図の例を図-1(c)に示す。)

(b) もし，全く同一に機能するアークが同一ノード間に2つ以上存在するなら，1つのアークを残して，残りのアークを消去する。

(c) 再びアークとノードを置き換えて評価グラフにもどす。

結果の評価グラフには，2つ以上の同一業務，同一機能のノードは存在していない。また，上記(a)～(c)の手順は，システムに同一業務，同一機能のプログラムが2つ以上存在しないことを確認するためにも使用可能である。



図—4 ソフトウェア設計手順

Fig. 4. Flow chart for software designing technique.

一方，(2)のケースは，連続する2つの処理を同一プログラムに統合する場合であり，プログラムの独立性が多少失われるが図-3(a)のようにA→Bの従属関係しかない場合には単純に可能である。しかし，図-3(b)のような場合には，A→B, C→Bは順序に関連を持って処理されるが，AとCの処理は一般に独立であり，A, B, Cを単純に同一プログラムに統合することはできない。したがって，A→Bのみの従属関係しかない時のみ同一プログラムに統合可能としている。

なお，(1)のケースは余分なプログラムの消去であるため，対応プログラム単体に対する評価は全く変わらず，プログラムの減少分だけ評価は良くなると考えられるが，(2)のケースはプログラムの構造に変化がおよぶため，プログラム単体の評価も大きく変化する可能性がある。

したがって，ここでは，(1)のケースを処置した後，(2)のケースについて，プログラムを統合しながら評価関数の値を計算し，評価関数の値が最適となるようなソフトウェアシステムを選択することによって最適ソフトウェアシステムを得る方法をとった。

本手法を用いて総合評価指標を最小とするシステムが最適システムとした場合の設計手法をフローチャートの形で図-4に示す。

3. 電力系統制御用ソフトウェアシステムの評価手法

ソフトウェアの評価規準は，従来，信頼性と効率に重点がおかれてきた。しかしながら，近年，ハードウェアの演算速度の向上，記憶装置の低価格化に伴い，評価規準は信頼性や効率よりもその使いやすさ作りやすさ，すなわち，保守性，更新性，理解性，テストの容易性，管理性といったものに変化してきている。このうち，特に電力系統制御用ソフトウェアに必要な評価概念を拾い上げると，表-1の抽象概念の欄に示すように表現できる。表-1は抽象概念を等価な表現に置き換え，その定量化要素とともに一覧表にしたものである。これら抽象的評価概念を定量的に評価する手法については，種々の方法が発表されてはいるが[3]，いずれも得失があり，現状において決定的手法はない。したがって，システムの評価規準として，設計・製作チームの構成とシステムの相異によって，それに合ったいかなる評価手法を用いても良く，ここでは抽象概念を，一般に良く知られた定量的評価手法を適用できる等価な表現に置き換えて定量的に評価する手法を試みた。したがって，表-1の等価な表現欄は抽象概念の内容を十分に表現できてはいないが，ソフトウェアを評価する一つの目安として考えてもさしつかえないと思われる。

もちろん，対象システムを評価する適当な評価手法があれば定量化要素の項目をその手法に置き換えることは可能であるし，抽象概念をさらに増すことも可能である。

表-1の場合，結局，ソフトウェアを評価する規準は，定量化可能と思われる表現で次の3つである。

(1) プログラムの結合度小・強度大

(2) プログラムの製作日数小

表—1　電力系統制御用ソフトウェアの評価

Table 1.　Evaluation of software for power system's control

抽象理念	等価な表現	定量化要素
更新性	① プログラム単体の構造が良い ② プログラムの独立性がある	① プログラムの強度大 ② プログラムの結合度小
信頼性	① 潜在するバグ数が少ない ② プログラムの独立性がある	① バグ数小 ② プログラムの結合度小
管理性	① 製作日数が短い ② プログラムの独立性がある	① ソフトウェア製作工数 ② プログラムの結合度小
理解性	① 単体プログラムのワード数小 ② プログラムの独立性がある ③ ソフトウェアの階層性 ④ ドキュメントの見易さ	① 製作日数・バグ数に帰着 ② プログラムの結合度小

(3) プログラム内に内在するバグ数小

プログラムの強度と結合度については，Myers らがその評価法を検討しており[4]，それを利用した。また，プログラムの製作日数と内在するバグ数については，従来から一般にいわれているように，プログラムの大きさの $m\,(m \geqq 1)$ 乗に比例する[5]として評価することにした。評価は (1), (2), (3) の項目を各々，製作工数または保修工数に換算して行った。

3.1 プログラムの強度と結合度

Myers らはプログラムのモジュール化手法を提案し[4]，その中で，モジュールの強度と結合度を定義し，強度ベクトルと結合度マトリクスとからグラフ理論と確率論を用いて以下のように評価マトリクスを導いている。

(1) 結合度マトリクス C

$$\boldsymbol{C}=\begin{vmatrix} C_{11} & C_{12} & \cdots\cdots & C_{1n} \\ C_{21} & C_{22} & \cdots\cdots & C_{2n} \\ \vdots & & & \vdots \\ C_{n1} & \cdots\cdots & & C_{nn} \end{vmatrix} \tag{1}$$

ただし，C_{ij} は，モジュール i と j の間の結合度によって定まる $0 \leqq C_{ij} \leqq 1$ の数値

(2) 強度ベクトル S

$$\boldsymbol{S}=[S_1,\ S_2,\ S_3,\ \cdots\cdots\ S_n] \tag{2}$$

ただし，S_i はモジュール i の構造によって定まる $0 \leqq S_i \leqq 1$ の数値

(3) 一次従属マトリクス E

$$\boldsymbol{E}=[E_{ij}] \tag{3}$$

で表わされるモジュール間の一次従属関係を表わすマトリクスで，その要素は次式で与えられる。

$$\left.\begin{array}{ll} E_{ij}=0.15\,(S_i+S_j)+0.7\,C_{ij} & (C_{ij} \neq 0,\ i \neq j) \\ E_{ij}=0 & (C_{ij}=0,\ i \neq j) \\ E_{ij}=1 & (i=j) \end{array}\right\} \tag{4}$$

(4) 完全従属マトリクス（評価マトリクス）F

$$\boldsymbol{F}=[F_{ij}] \tag{5}$$

で表わされるモジュール間の完全従属関係を表わすマトリクスで，その要素は次式で与えられる。

$$\begin{aligned} F_{ij}=&\sum_{l_{i_1}}^{l_m} P_{ij}(l_{i_1})-\sum_{\substack{l_{i_1},l_{i_2} \\ (l_{i_1}<l_{i_2})}}^{l_m} P_{ij}(l_{i_1})\cdot P_{ij}(l_{i_2})+\cdots\cdots \\ &+(-1)^{k-1}\sum_{\substack{l_{i_1},l_{i_2}\cdots l_{ik} \\ (l_{i_1}<l_{i_2}<\cdots<l_{ik})}}^{l_m} P_{ij}(l_{i_1})\cdot P_{ij}(l_{i_2})\cdot\cdots\cdots\cdot P_{ij}(l_{ik})+\cdots\cdots \\ &+(-1)^{m-1}P_{ij}(l_1)\cdot P_{ij}(l_2)\cdot\cdots\cdots\cdot P_{ij}(l_m) \end{aligned} \tag{6}$$

ただし，$P_{ij}(l_k)$ はモジュール i から j への結合が他のモジュールを介して結合する場合も含めて m 通りあるとき，第 k 番目の結合にそって一次従属マトリクス $\boldsymbol{E}$ の要素を掛け合せたものであり，次式で与えられる。

$$P_{ij}(l_k)=E_{ia_1}\cdot E_{a_1a_2}\cdot\cdots\cdots E_{ak\,j} \tag{7}$$

ここで

$$FD=\frac{1}{n}\sum_{i}^{n}\sum_{j}^{n}F_{ij} \tag{8}$$

を定義すると，これは，n 個のもジュールがあるとき，そのうちのある1個のモジュールが変更されたとき，その影響がおよぶモジュールの個数の期待値を表わす。

Myers らによる以上の関係から，モジュールをプログラムと読み換えて次の関係式を導く。

$$FC=k_0\frac{1}{n-1}\quad(FD-1) \tag{9}$$

(9)式は，今，簡単のため，あるプログラムに変更があったとき，他のプログラムへおよぶ変更について，1個のプログラムを変更する日数（人工）がプログラムの分割の個数に反比例して影響を受けると仮定した場合の，あるプログラムに変更があったときの他プログラムの変更に要する日数（人工）を表わしている。以下，プログラムの結合度と強度を表わす評価指標として，(9)式の値を用いる。

3.2 製作工数

全体システムが n 個のプログラムより成り，各プログラムのワード数を N_1, N_2, … N_n とし，製作工数がプログラムワード数の m 乗に比例すると仮定すると，プログラムの製作工数 Q_P は

$$Q_P=\sum_{i=1}^{n}k_1\cdot N_i^m \tag{10}$$

また，プログラム i とプログラム j の相互関係に要する製作工数を $Q_{C_{ij}}$ とし，他プログラムを介する場合の結合度を無視して，C_{ij} に比例すると仮定すると，プログラムの相互関係に関する製作工数 Q_C は

$$Q_C=\sum_{i=1}^{n}\sum_{j=1}^{n}Q_{C_{ij}}=\sum_{i=1}^{n}\sum_{j=1}^{n}k_2\cdot C_{ij}\qquad(i\neq j) \tag{11}$$

さらに，データとプログラムの結合に関する製作工数 Q_I は，一定の基準工数とデータ量に比例する工数に分けられると仮定すると

$$Q_I=\sum_{i=1}^{n}k_3\cdot(D_{I_i}+D_{O_i})+nB \tag{12}$$

ただし，D_{I_i}；プログラム i への入力データ数

D_{O_i}；プログラム i からの出力データ数

B　；プログラムとデータの結合に要するプログラム1個あたりの基準工数

以上から，全体の製作工数 W は次式で与えられ，以後，製作工数の評価指標としてこれを用いる。

$$W = Q_P + Q_C + Q_I \tag{13}$$

3.3 バグ修正に要する工数

プログラムにおけるバグ発生数 P_P はプログラムワード数の b 乗に比例すると仮定すると

$$P_P = \sum_{i=1}^{n} k_4 \cdot N_i^b \tag{14}$$

製作工数の場合と同様に，プログラム相互関係によるバグ発生数 P_C と，プログラムとデータの結合に関係するバグ発生数 P_I は，それぞれ

$$P_C = \sum_{i=1}^{n} \sum_{j=1}^{n} k_5 \cdot C_{ij} \tag{15}$$

$$P_I = \sum_{i=1}^{n} k_5 \cdot (D_{I_i} + D_{O_i}) \tag{16}$$

したがって，1つのバグを修正するのに要する平均工数を k_7 とすると，バグ修正に要する平均工数 P_D は次式で求めることができ，以後，バグ修正に要する人工の評価指標として，これを用いる。

$$P_D = k_7 \cdot (P_P + P_C + P_I) = k_7 \cdot P_B \tag{17}$$

ただし，　$P_B = P_P + P_C + P_I$

3.4 システムの製作・変更に要する全工数

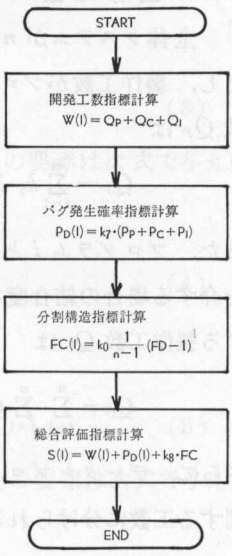
今，システムの製作工数を製作工数と製作時に発生したバグを修正するのに要する工数と仮定し，また，将来，k_8 回の変更が予想されると仮定すると，製作工数と将来の変更に要する工数も含めた全工数 WT は

$$WT = w_1 \cdot W + w_2 \cdot P_D + w_3 \cdot k_8 \cdot FC \tag{18}$$

ただし，　w_1. w_2, w_3 は重み係数

で与えられる。したがって，ソフトウェアシステムを総合評価指標である全工数 WT, すなわち(16)式が最小となるよう設計すれば良いことになる。

以上の評価手順を図-5のフローチャートに示す。



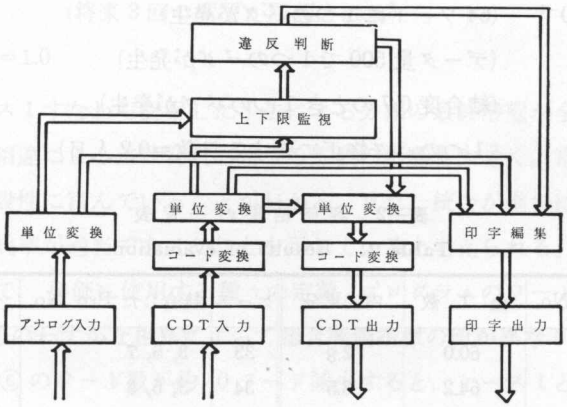
図—5　評価手順

Fig. 5. Flow chart for evaluation.

4. 簡単な例題と結果の考察

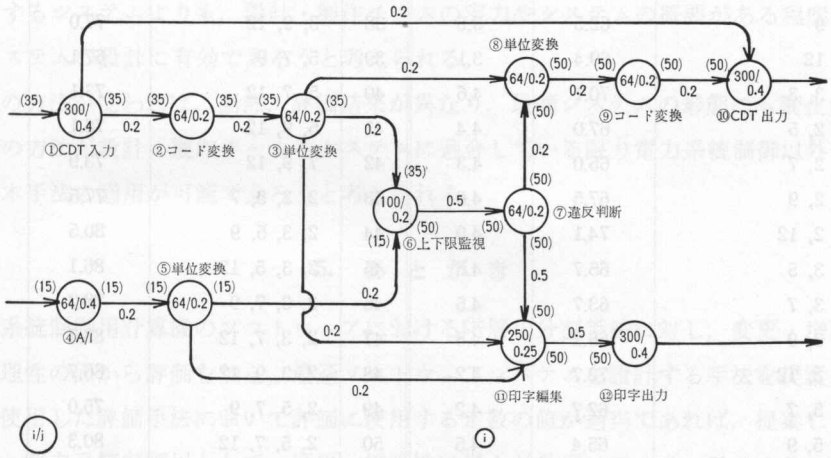
前稿の例題で得られた図-6のソフトウェアシステムに対して，前節に示した評価を行い，最適システムの設計を試みる。

図-6のシステムを評価グラフの形に描くと図-7のように表わすことができる。図-7のプ



図—6　ソフトウェアの分割例

Fig. 6.　Example of software decomposition.



注）i/j；i＝プログラムワード数，j＝プログラム強度　　i；プログラム番号
(i)；データ数（→の先は入力，根は出力）　　$\tilde{t}$；プログラムの結合度

図—7　評価グラフ

Fig. 7.　Evaluation graph.

ログラムの強度と結合度は，Myers が文献[4]に示している値を参考にして適当に定めた。

各定数を以下のように仮定し，統合可能なプログラムを順次統合しながら総合評価指標（全工数）を算出した結果を表-2に示す。

$m = 1.5$

$b = 2.0$

$k_0 = 30.0$　　（全プログラムに影響が及んだとき30日）

$k_1 = 6.9 \times 10^{-5}$　　（128ワード製作するのに1人日）

$k_2 = 1.0$　　（結合度0.5のとき0.5人日の打合せが必要）

$k_3 = 0.002$　　（データ量500の製作に1人日）

$k_4 = 2.44 \times 10^{-4}$ (64 ワードに 1 つのバグが発生)

$k_5 = 0.002$ (データ量 500 で 1 つのバグが発生)

$k_6 = 1.43$ (結合度 0.7 のとき 1 つのバグが発生)

$k_7 = 0.2$ (1 つのバグ修正に要する日数＝0.2 人日)

表—2 評価結果の一覧表

Table 2. Results of evaluation

ケース	統合した Pro. No.	全工数	内変更分	ケース	統合した Pro. No.	全工数	内変更分
1		60.0	2.8	33	3, 5, 7	67.5	5.8
2	2	64.2	3.5	34	3, 5, 9	70.4	6.2
3	3	62.8	3.7	35	3, 5, 12	76.4	5.4
4	5	62.1	3.4	36	3, 7, 9	68.3	6.1
5	7	60.1	3.4	37	3, 7, 12	74.2	5.3
6	9	62.5	3.6	38	3, 9, 12	77.0	5.7
7	12	69.4	3.1	39	5, 7, 9	67.1	5.7
8	3, 3	70.1	4.6	40	5, 7, 12	73.1	4.9
9	2, 5	67.0	4.4	41	5, 9, 12	76.1	5.3
10	2, 7	65.0	4.3	42	7, 9, 12	73.9	5.2
11	2, 9	67.5	4.6	43	2, 3, 5, 7	77.5	7.6
12	2, 12	74.1	4.0	44	2, 3, 5, 9	80.5	8.0
13	3, 5	65.7	4.6	45	2, 3, 5, 12	86.1	7.1
14	3, 7	63.7	4.5	46	2, 3, 7, 9	78.2	7.9
15	3, 9	66.2	4.8	47	2, 3, 7, 12	83.7	6.9
16	3, 12	72.7	4.2	48	2, 3, 9, 12	86.7	7.3
17	5, 7	62.7	4.2	49	2, 5, 7, 9	75.0	7.6
18	5, 9	65.4	4.5	50	2, 5, 7, 12	80.3	6.6
19	5, 12	71.9	3.9	51	2, 5, 9, 12	83.6	7.1
20	7, 9	63.3	4.5	52	2, 7, 9, 12	81.2	6.9
21	7, 12	69.8	3.8	53	3, 5, 7, 9	74.1	8.0
22	9, 12	72.4	4.1	54	3, 5, 7, 12	79.3	6.9
23	2, 3, 5	74.1	5.9	55	3, 5, 9, 12	82.6	7.5
24	2, 3, 7	72.0	5.8	56	3, 7, 9, 12	80.2	7.3
25	2, 3, 9	74.5	6.1	57	5, 7, 9, 12	78.8	6.8
26	2, 3, 12	80.8	5.4	58	2, 3, 5, 7, 9	86.7	10.7
27	2, 5, 7	68.7	5.5	59	2, 3, 5, 7, 12	91.3	9.4
28	2, 5, 9	71.5	5.9	60	2, 3, 5, 9, 12	94.9	10.1
29	2, 5, 12	77.7	5.1	61	2, 3, 7, 9, 12	92.2	9.8
30	2, 7, 9	69.4	5.8	62	2, 5, 7, 9, 12	88.9	9.5
31	2, 7, 12	75.4	5.0	63	3, 5, 7, 9, 12	88.2	9.9
32	2, 9, 12	78.2	5.4	64	2, 3, 5, 7, 9, 12	104.8	13.9

(単位： 人日)

$k_8 = 30$　　（将来3回の変更が予想される）

$w_1 = w_2 = w_3 = 1.0$

表-2から，ケース1すなわち提案した手法による分割の最終形態が全工数において最小であり，また，分割の相違による差は変更に要する工数の影響も大きく，電力系統制御において最も重要な変更・増設性に富んでいることがわかる。また，統合が進むにしたがって評価が悪くなるのは分割時の規準が適当であったことを示していると考えられる。

評価手順において，評価に使用する種々の定数（プログラムのワード数や強度・結合度，係数，重み係数等の値のわずかな相異によって総合評価指標の値が異なり，たとえば，表-2において，プログラム⑥のワード数が約40ワード減少すると，ケース1とケース5の総合評価指標の値が逆転し，ケース5が最適となる。したがって，評価のための定数や重み係数の値の決定には十分な注意を払い，実績のある値を利用する必要がある。そのため，本手法は全く新たに設計するシステムよりも，設計・製作チームの実力やシステムの概要がある程度わかる経験あるシステムの設計に有効であろうと考えられる。

評価の方法が変われば，当然，評価結果が異なり，最適システムの形態にも変化が生じるが，評価の方法が設計・製作チームやシステムに適合している限り電力系統制御以外の分野においても本手法の適用が可能であろうと考えられる。

5. あとがき

電力系統制御用計算機のソフトウェアにおける階層化分割手法に対し，変更・増設性，信頼性，管理性の面から評価を加え，最適ソフトウェアシステムを設計する手法を提案した。その結果，使用した評価手法において評価に使用する定数の値が適当であれば，提案した設計手法により，電力系統制御用として，変更・増設性に富んだ最適ソフトウェアシステムの設計がなされることが確認できた。

また，本手法は，評価手法や評価規準を対象システムに合せることによって電力系統制御システム以外の制御システムにも適用できると考えられる。

なお，本手法は電子計算機による自動評価が可能であり，電子計算機援用設計（CAD）を採用することによって，ソフトウェア設計の省力化が期待できる。

参考文献

1) 奈良・辻：北見工大研報，第9巻，第2号（昭和53年4月）.
2) 奈良・辻：電気学会情報処理研究会資料，IP-77-62（昭和52年12月）.
3) 例えば宮本・東谷：bit 第117号（昭和53年1月），第118号（昭和53年2月）.
4) Myers，久保他訳：高信頼性ソフトウェアー複合設計（1976）近代科学社.
5) 例えば藤野・紫合：信学誌 Vol. 59, No. 1（昭和51年1月）.