ソフトウェア工学 24－4
（1982. 6. 4）

ソフトウェア要求定義からプログラムの構造設計まで

松本吉弘
（東京芝浦電気株式会社）

1、まえがき

大規模なソフトウェアの設計におけるプログラムの巨視的構造の問題を扱っている。ここで対象として扱っているソフトウェアは、個々が数メガバイトのメモリと数百メガバイトの補助メモリをもつリアルタイム用計算機複数台から成る複合システムに装填されるようなものである。以下の記述を行うに至らしめる問題というのは、各計算機にどのようにソフトウェアを分担して実行せしめるかという設計上の判定、モジュール割付け，…など初期段階の構造設計に関係する。

目的のソフトウェアをいかに分割して、各リソースに割付けるのが合理的であるか；たとえば、もっとも分り易い例では、交信する外界システムに従って割付けるのがよいのか（タテワリ）、ソフトウェアの実施する機能によって割付けるのがよいのか（ヨコワリ）という問題がこれに含まれる。

このような割付けを決定するためのファクタは数多く考えられる。ユーザの基本的思想，馴れ，保守上の配慮から始まり、複合して作動する際の性能上の問題まで広範囲に亘っている。ここで述べるような手法はその決定のためのファクタのひとつにしかなり得ないが、ソフトウェアを合理的に構成していく上に、大きな助けになっているものである。

2、従来手法との関連

ソフトウェア設計に関する手法の発表は要求空間に関するものと、実現空間に関するものとに大別される。図1に示すように要求空間では要求分析、要求定義がなされ、その後これが実現空間に移され、実現するための要素決定、要素の組合せ構成が行われる。要求空間，実現空間それぞれの中において用いられる手法や技法についてはサーベイペーパ[1][2]などによってもわかるように多くのものがある。要求空間と設計空間を結ぶ図1の二重矢に対応する手法については、明白に触れた文献があまりない。この原稿は二重矢に対応した筆者の今までの考察[3][4]を延長したものである。

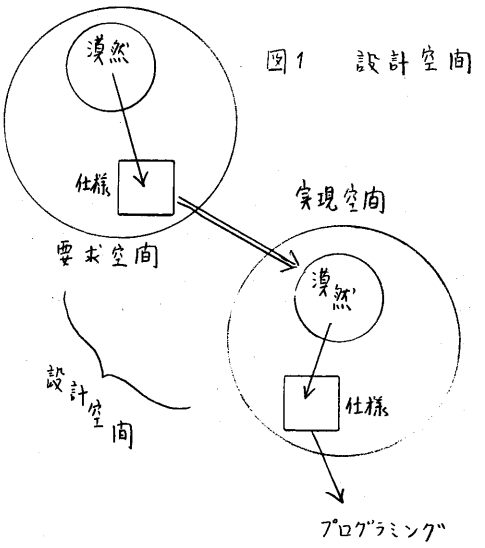

図1　設計空間

3、要求空間における最終形式

要求空間における最終形式はつぎの要素から成立つと考えられる。

イ、要求単位（やらねばならないこととその範囲）
ロ、要求単位間の関係
ハ、制約
これらが適当な仕様言語，または図式によって表わされたものが要求の最終形式である。

4、実現空間における最初の表現

実現空間に入って設計者が最初に手をつけることは、上で定義されたイ、ロ、に従って、最後に生成されるであろうコードが実現すべき作用単位（機能とよぶ）とデータ単位を割出し、機能とデータ単位との関係を創造することである。機能単位とデータとの関係づけを行う手法は数多く提案されている。[5]
図2は一般的な手法で表現した機能／データ関連図の1例である。実線はデータ、点線は制御を表わしている。X_1, X_2, X_3 は機能単位を表わし、D_1 はデータ集合（データベース，またはファイルの作用）を表わしている。

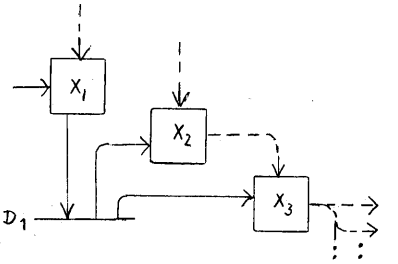

図2　機能／データ関連図例

この例では X_1 は外部プロセス変数の走査（スキャン）作用，X_2 は外部プロセスの状態同定作用，X_3 は外部状態に対応したタスクの選択作用を表わしている。
図2は〝外部プロセスの状態に対応した機能を選択せよ〟という要求単位 A_1 に従って創造されたもので、図3に示すような要求図式に対応している。
図3は要求空間内で作られ，図2は実現空間内で作られたものである。図2と図3を対比してみると、両者の間には箱や矢の型に変化がみられるが、図2が図3の各オブジェクトの詳細化によって作られたかの如き態様を示す。数多くの例を扱った経験から、要求図式と、機能／データ関連図の間には何らかの変換則をもって関連づけ得ることを暗示する相似性があると考えられる。

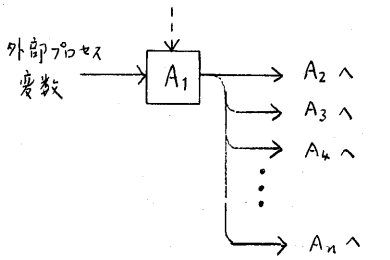

図3　図2をもたらした
要求図式の例

5、要求図式から機能／データ関連図への変換

図1の二重矢に対応する活動、すなわち要求空間から実現空間への変換は真に人間の知的な思考を必要とする部分で、これこそもっとも〝設計者〟の真価を問われる部分である。このような活動を形式化する努力は必然的に人工知能，知識工学分野における蓄積を利用することになる。Programmer's Apprentice[6]，

Automatic Programming[7]などの研究はこういった意味で参考になる。要求図式から機能／データ関連図への変換を支援するシステムについて触れてみよう。このシステムはつぎのような機能をもつ。

イ、要求と要求図式の型を記憶する機能

ロ、変換オペレータを決定するためのルールと、マッチング機能

ハ、変換オペレーションを行う機能

要求と要求図式には数多くの型が考えられるが、それぞれの応用に即して特定化できる。ルールやオペレータについても同様である。なかなか一般的な変換システムを作ることは容易でないが、もっとも生産性を上げたい特定の応用に限って、知識ベースを作り上げていくことはそれほど難しいことではない。

6、プログラムの巨視的構造（または大構造）の決定

ここで巨視的構造といっているのは、モジュール（パケージなど），タスク相互間の構造的関係のことである。早い段階でプログラムの合理的構造を定めて、プログラマの分担を決めるということは重要なことである。とくに複数の計算機から成る複合システムにおいて、プログラムの割付けを決定する場合には、この巨視的構造が大きな役割をもつ。図3から図2のような変換が行われるとすれば、プログラムの巨視的構造は要求のもつ構造と一致するはずである。もし、それが現実に可能とすれば、プログラムシステムはより外から見て理解しやすく保守しやすいものになる。しかし、多くの場合そうならないのは、図2のX_1，X_2，X_3と同じ作用を示す機能が、他の部分にも数多く現われ、これを統合せねばならないからである。このためにプログラムの巨視的構造は要求のそれと異なるのが一般的である。既経験のシステムや、小規模なソフトウェアではプログラマが初めから巨視的構造を頭に描けるから論外である。

構造をきめる上で判断のよりどころになる因子は、機能間の制御パスとデータ参照パス、それぞれのパス数，1パス内の伝達量，パスの発生頻度である。機能／データ関連図からプログラム構造を割出す経験的手法[5]がある。筆者は複雑にからみ合う機能／データ関連図からクラスタリングという方法で構造を割出す試みを行っている。これによれば、図4(a)，図5(b)のような機能／データ関連図はそれぞれ図4(b)，図5(b)に示すような構造図に変換される。[8]

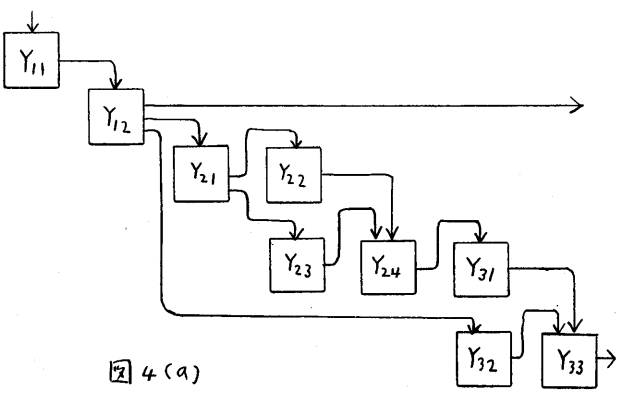

図4(a)

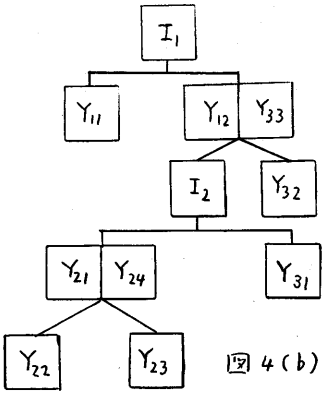

図4(b)

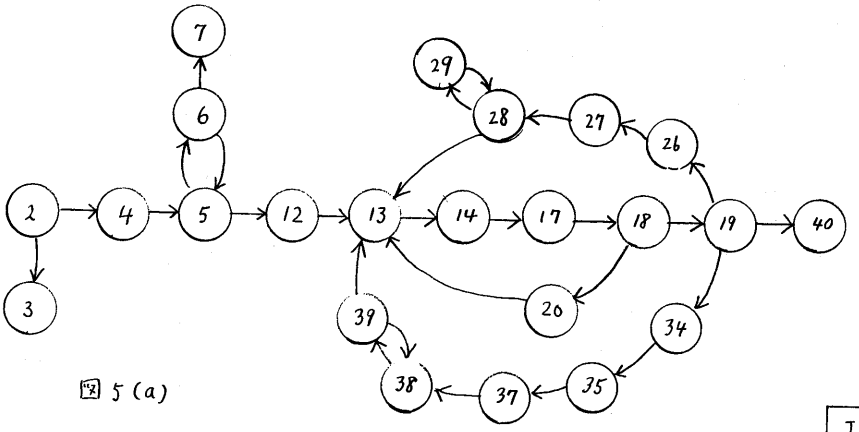

図5(a)

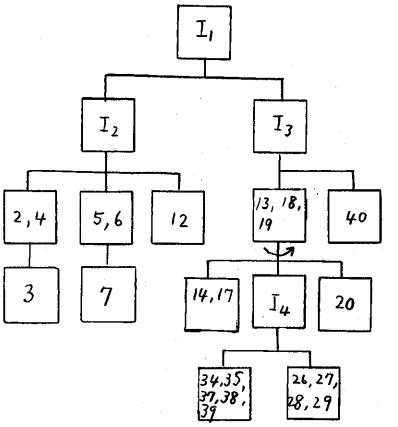

図5(b)

図4の例は並列に実行可能なタスク（Y_{22}, Y_{23}, および, I_2, Y_{32} はそれぞれタスクとなし得る）を含む例で、図5はくり返し，または再帰を含む例である（⤳のかったブロックは繰返す）。図5(a)の表現では、丸が作用を表わし、有向矢がデータを伴う制御の流れを表わしている。

図4(b), 5(b)のような構造図は設計者がモジュール（またはタスク）の割付け，構成を決定する際に参考となり得る。ある枝に接続された葉に相当する作用（機能）をまとめていとつのモジュールで実現するように考えることができる。たとえば、図5(b)の構造図から、図6のようなモジュールの構造を決定することができる。モジュールM_1では図5(b)のI_2グループの作用を実現する能力をもち，M_3は40の作用を、さらにM_2ではI_3グループの40以外の作用を実現する能力をもつものとする。

このようにして、機能／データ関連図は、モジュールの大構造にまで、展開できる。

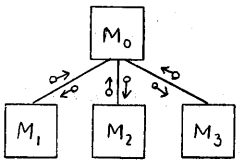

図6　モジュール構造図の例

6、むすび

ソフトウェアに対する要求定義からプログラムの巨視的構造（大構造）を創造するまでの変換を一貫した手法で行う試みについて述べた。この試みはまだ完成したものではないが、一部は筆者らが開発したSWBシステム[9]の中で生きて活動している。

引用文献

[1] Ramamoorthy,C.V., etal., Tutorial: Software Methodology, IEEE Cat.No. EHO 142-0, IEEE Computer Society, pp.43-164

[2] Riddle,W.E., etal., ACM SIGSOFT, Software Engineering Notes, Oct. 1978, pp.7-11

[3] Matsumoto,Y., etal., Proceedings of COMPSAC'80, IEEE Computer Society, pp.259-266

[4] Matsumoto,Y., Computer Science & Technologies 1982, Editor T.Kitagawa, OHM-North-Holland, 1982, pp.175-192

[5] Bergland,G.D., Computer, IEEE Computer Society, Oct. 1981, pp.13-37

[6] Hewitt,C.E., etal., IEEE Trans. on Software Engineering, Vol. SE-1, No. 1, March 1975, pp.26-45

[7] Barstow,D.R., Artificial Intelligence, North-Holland, 1979, pp.73-119

[8] Matsumoto,Y., A clustering algorithm applied to the software development, submitted to COMPSAC 82, IEEE Computer Society

[9] Matsumoto,Y., etal., SWB System: a software factory, Software Engineering Environments, editor H.Hunke, North-Holland, 1981, pp.305-318