プログラムの機能表現から構造表現への自然な変換法について

酒井 三四郎
（静岡大学電子科学研究科）
落水 浩一郎
（静岡大学工学部）

1. はじめに

我々はソフトウェア開発保守支援システムPDBを開発した。[1][2] 本研究はその設計支援機能の充実をねらったものである。本報告は与えられた機能要求（要求記述）を，機能の分析と詳細化を行ない，プログラムのモジュール構造に変換する方法について述べている。

本設計法の概要を図1に示す。要求記述とは，利用者とのインタフェースである画面，報告書の仕様，コマンドとその機能などを定めたもので，本設計法では別の方法で与えられるものとする。

機能設計とは，与えられたプログラム全体の機能を，2章で述べる基準で分解し，図式表現する過程である。

構造設計とは，上述の図面を基に機能構造をモジュール階層構造に変換する過程である。3章で述べる。

我々の設計法は以下の特徴を有する。

(1) 機能の分析と詳細化は，データの流れに基づいて行なっており，しかも機能間の種々の相互関係を図式を用いて表現しているので理解しやすい。

(2) 設計者の経験とか直感力によって差が生じないように，ある手順にそってなるべく機械的に実施させるので，誤りの発生を抑制する効果をもつ。

(3) 機能表現中の要素および要素間の関係と構造表現中のそれとの間に分かりやすい対応があり，それらは保守性の高い設計図面の役割を果たしうる。

現在提案されているプログラム設計法の一つ，複合設計[6]では，主要なデータの流れだけに着目するために特徴(3)が満たされにくい。また，ジャクソン法[3]は，データ構造という具体的なものに着目するために，機能定義から構造表現への変換はかなり容易である。ただ，モジュール間のインタフェースの設計には適用しにくいように思われる。本設計法はこれらの点を改善している。

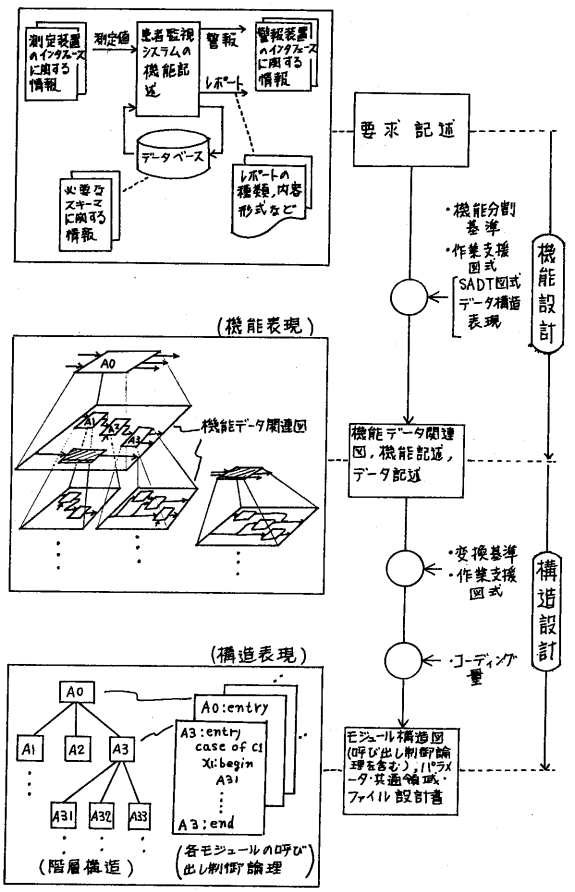


図1 設計法の概観

2. 機能設計

機能設計とは，以下に示す機能間の関係を定める作業である。

i) 階層内関係
- a) 順序付けられる関係
- b) 選択される関係
- c) 繰返される関係
- d) 独立の関係

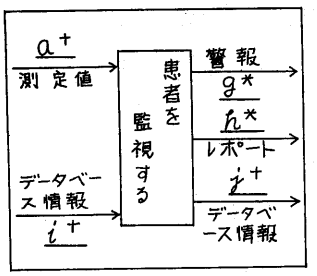


図2 例題の機能設計(1)

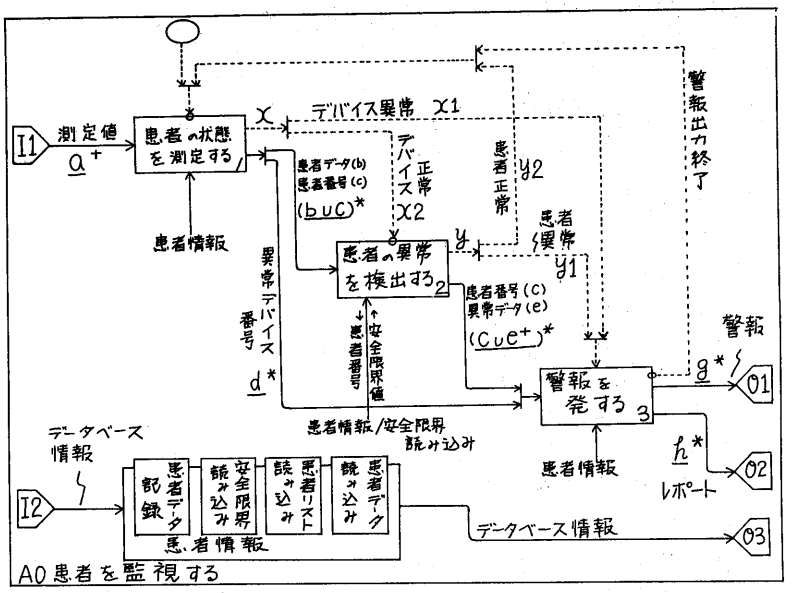


図3 例題の機能設計(2)

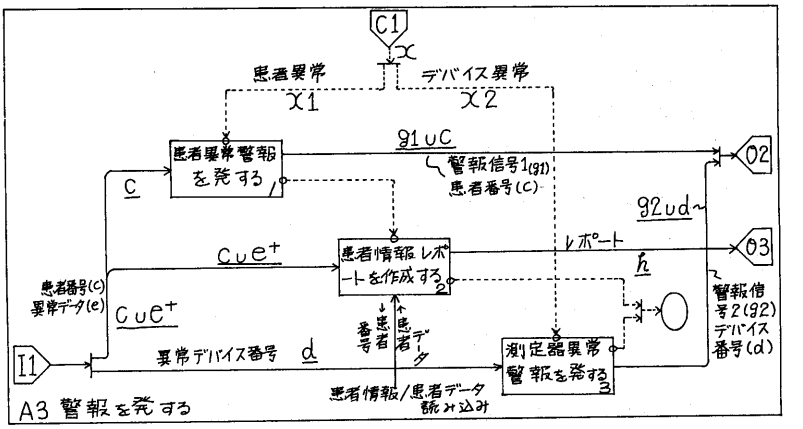


図4 例題の機能設計(3)

ii) 階層間関係(機能の上位, 下位の関係で, 一般には木構造). この時, 以下の関係が生ずる.

a) 上位機能は下位機能間の動作の順序, 選択, 繰返しなどを制御する.

b) 上位機能は下位機能の入出力データへのアクセスを制御する.

本稿では, 方法論の体系全体を説明するのではなく, 例題に基づいて説明する. 例題としてConstantineの患者監視システムの設計問題をとりあげる.

機能設計とは, たとえば図2が与えられた時, 入出力データの構造と, ボックスに示されている機能の分析に基づいて, 図3中の3つのボックスを得る. 次にそのボックス間の順序関係, 新たに導入されたデータの構造を定める. さらに, 図3中の右下隅のボックスに注目して, 同様の手順で図4を得るという作業である.

2.1 作業図式

本局面で用いる図式は(i)機能データ関連図, (ii)機能記述, (iii)データ記述の3種類である. (i)はSADT[4][5]の動作分解の図式を基にする. (iii)は機能データ関連図中のボックスについての付加的な説明を記述したものである. (iii)は機能データ関連図中のアロー上を流れるデータに関する説明, 構造等を記述したものである. 構造に関してはジャクソンのデータ構造図[3]を基にする.

ここで, 主に機能データ関連図で用いる表記法について説明する.

【データ構造の表現】

一般に詳細化の対象ボックスは図5のような構造を有している. C,X,Yは構造を持っており, その

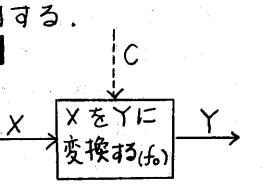


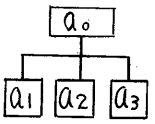
図5 対象ボックスの一般形

表記法を説明する

表1 データ構造の記号表現

	型	表記法と説明(例)
1	基本型	それ以上分解する必要がなく，どんな部分ももたない基本的要素．英子文字または添字付きの英子文字で表わす．(a，a_1)
2	連接	abc，$a_1a_2a_3$ など．
3	繰返し	a^*，a_1^+ など．
4	選択	avbv— など．「—」は空を表わす．
5	合併	aubuc など．

表1の1～4についてはジャクソンの図的表記法を記号表現したものである．5は個々の要素の集合の意味で，図的表記法を図6に示す．図的表記と説明はデータ記述に，記号表現は機能データ関連図のアーク上に書く．



$a_0 = (a_1 \cup a_2 \cup a_3)$

図6 合併の図的表記法

【処理単位の表現】

機能データ関連図中のアークはデータの流れを意味しているが，各機能が1回に処理する単位を明らかにすることによって，機能間の関係が明確になる．さらに，その機能を詳細化する場合には，処理単位内の構造だけに注目すればよい．処理単位の表現は，上述したデータ構造の表現中で，単位となる部分に下線を引いて表わす．

(例) $\underline{a^*}$，$\underline{a}^*$，$(\underline{a_1a_2a_3})^*$ など．

【破線のアローの意味】

破線のアローは機能間の起動順序を表わすために用いる．この時，アローとボックスの接点に"o"印が付く場合と付かない場合がある(図7)．(1)の場合は単に「A1を起動する」，「A1が終了した」ということを意味するだけで，A1の内部には影響を与えない．一方，(ロ)の場合は，(1)の意味に加えてA2を詳細化したボックスの起動に関しても影響し，逆に，それらのボックスからA2の外のボックスの制御に影響を与える．

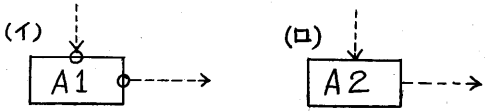


図7 破線のアローと"o"印の意味

たとえば，図3のボックス「警報を発する」の制御入力は図7－(ロ)の型で，そのボックスを詳細化した図4において，最初に起動されるボックスを制御する．

表2 制御アローの意味

	表示	説明
1	○----→□	1番初めに起動されるボックスを示す．
2	□----→○	最後に起動されるボックスを示す．

また，表2はボックス間の起動の順序を完全に定めるために用いる．外部から(外部への)制御アローによってそれが明らかな場合は省略する．

【アローの分岐表現】

機能間を流れるデータの分岐，統合を表現するために表3の表記法を用いる．1～6はSADTの表記法に基づいている．

表3 アローの分岐表現

	表示	説明
1	AvB → A, B	OR分岐(排他的)
2	A, B → AvB	OR統合(排他的)
3	A → A, A	分岐
4	A, A → A	統合
5	C=(AUB) → A, B	展開
6	A, B → C=(AUB)	たばね
7	C=(AB) → A, B	連接分岐
8	A, B → C=(AB)	連接統合

2.2 作業手順

以下に述べる(1)～(3)の手順に従って下位階層方向に機能を分割し，新しい階層内で機能とデータ間の関係を定める作業を繰返す．

(1) 対象ボックスのアークと処理単位の指定に基づいて，下位の新しい階層を認識する．

たとえば図2を対象ボックスとする．「患者を監視する」という機能の入出力データの構造と処理単位が，$\underline{a}^+$，$\underline{g}^*$ などのように定義されている．よって，下位

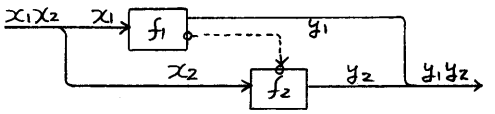
の新しい階層は，a^+，i^+を入力として，g^*，h^*，f^+を出力する機能の集まりとして認識される．このようにして，図3の境界アローの存在などの周辺部が定まる．"▷"印が対象ボックスと下位の新しい階層とをつないでいる．

(2) 「要求記述」中の機能記述および以下に述べる基準に基づいて機能を分割し，機能間の関係を定める．

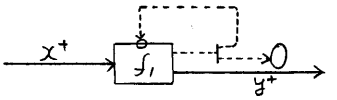
【 データ構造に基づく分割 】

入出力データの構造によって機能と，機能間の関係がどのように定まるかを示す．図5を対象ボックスとする．

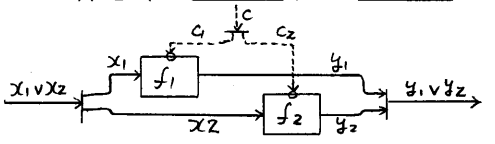
a)連接構造 (X: x_1x_2 , Y: y_1y_2)



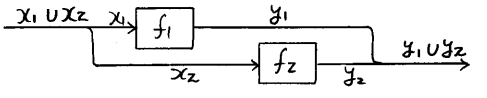
b)繰返し構造 (X: x^+ , Y: y^+)



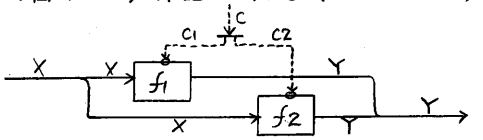
c)選択構造 (X: $x_1 \vee x_2$, Y: $y_1 \vee y_2$)



d)合併構造 (X: $x_1 \cup x_2$, Y: $y_1 \cup y_2$)

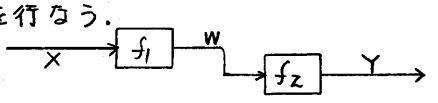


e)制御データの選択構造 (C: $c_1 \vee c_2$)



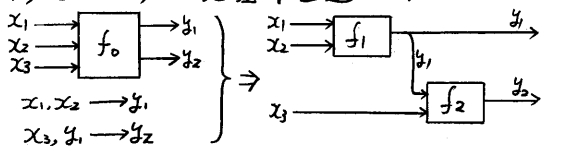
【 入出力データ間に構造の対応がない場合の機能分割 】

入出力データ間の構造に対応がない場合に，中間データを用いた段階的変換を行なう．



【 入出力データが複数の場合の分割 】

データ間の対応によるスレッドを考えて，スレッド上にボックスを配置することで，ボックス当りの入出力データ数を減少させて，上記基準を適用する．



【 抽象データの認識による分割 】

詳細化の際に，対象ボックスの入出力データの内，あるものは抽象データとして認識される．

たとえば，図2において「データベース情報」を抽象データ「患者情報」と認識して図3の下部分のボックスを得る．

ここでは，抽象データの操作として4つ認識されている．このボックスに関する詳細化は，図2，図3，図4という詳細化の階層構造とは別にもう1つの階層構造を形成する．

以上，手順(2)の分割基準を示したが，例題に適用してみる．図2では入出力データの処理単位が繰返し構造である．このことから，分割後の機能間に繰返し構造が現われることがわかる．また，図3のボックス「警報を発する」においては入力データと制御アークに選択構造がある．このことから，分割後の機能間に選択構造が現われることがわかる．

(3) 新しい階層内での定義を完結させる．

(2)で新たに導入したデータの構造，処理単位，機能間の起動の順序などを完全に定める．そして，その内の1つのボックスを取り出し，対象ボックスとして以上の手順を繰返すことになる．

3. 構造設計

構造設計とは，1枚の機能データ関連図を基に，モジュールの1段の呼び出し関係に変換すること

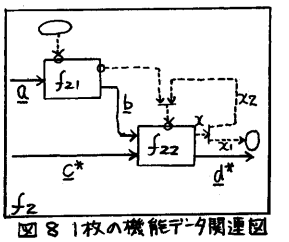


図8 1枚の機能データ関連図

を繰返す作業である．

たとえば図8においてf_2を親モジュールに，f_{21}，f_{22}を子モジュールに変換し，破線のアローを親モジュール内での呼び出し制御のアルゴリズムに，実線のアローをパラメータ等のインタフェースに変換する作業である．

3.1 設計手順

機能データ関連図をトップダウンに取り出しつつ，(1)，(2)の作業を反復する．最後に(3)に示すような手直しを行なう．

(1) 作業用紙に機能データ関連図の略図を作り，各アーク上にデータ構造と処理単位を書き込み，インタフェースの媒体を表4に示す記号で記入する．

表4 インタフェース媒体の表現

	記号	意味
1	●→	パラメータ
2	⊟→	共通領域
3	■→	ファイル

表4に示すように本設計法では，インタフェースの媒体としてパラメータ，共通領域，ファイルの3種を考える．

次に各アローにラベルを与える．略図において，ボックスと接しパラメータ，ファイル，共通領域として定められたアローに対して，それぞれP_n，F_n，D_n ($n=1,2,\cdots$) をつける．ただし制御アローのうち，パラメータが割りあてられたものについてはC_nをつける．

たとえば図9は図3の機能データ関連図に対して作成されたものである．まず，境界アローに関しては1つ上の階層における作業によって媒体が定まっているので，それをボックスと境界アローの交点に付ける．図9の場合はすべてファイルである．次に他のアローについて定める．基準としては，マイヤーズのモジュール結合度[5]の概念を参考にし，なるべくスタンプ結合またはデータ結合とする．共通結合または外部結合については，一度初期設定された後，変更されることのないデータで広範囲に参照されるものに限って共通領域を割りあてることにする．

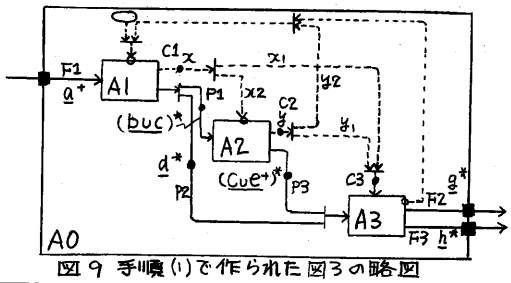


図9 手順(1)で作られた図3の略図

(2) (1)で作られた略図に基づいて，モジュールの呼び出し制御とデータへのアクセス法を定める．

略図とモジュール階層構造の間には，図10に示す対応がある．

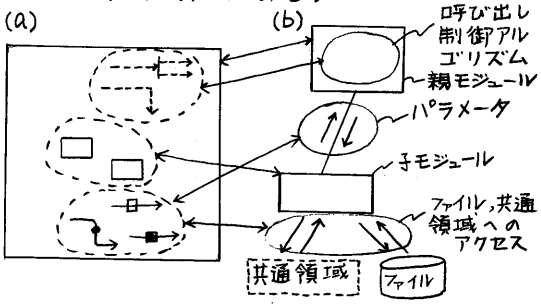


図10 略図とモジュール階層構造の対応

図10において，(a)の外側の枠すなわちノードを親モジュールに，内側の枠すなわちボックスを子モジュールに，それぞれ対応させる．破線のアローは親モジュール内の呼び出し制御アルゴリズムに，実線のアローはそれに付けられた記号によって，それぞれパラメータ，ファイル，共通領域へのアクセスに対応させる．

以上の情報を親モジュールの論理を記述する言語で表現する．

```
A0:entry(main; fi:F1; fo:F2,F3)
      loop
        A1(po:C1,P1,P2; fi:F1)
        case of C1
          x2:begin
               A2(pi:P1; po:C2,P3)
               case of C2
                 y1:A3(pi:P3,P2,C3; fo:F2,F3)
                 y2:——
               end
             end
          x1:A3
        end
      end
A0:end
```

図11 親モジュールにおける制御論理の言語表現

たとえば，図11に示すような言語表現である．繰返し，選択，連接を表わす構文がある．その他，入口，出口，モジュール呼び出しの文があり，入口とモジュル呼び出しの文には，インタフェース情報を図12のような記号を用いて付ける．また，例には現われていないが，goto文とラベルを用いることもある．

(pi：入力パラメータの並び，… ；
po：出力パラメータの並び，… ；
fi：入力ファイルの並び，… ；
fo：出力ファイルの並び，… ；
di：入力共通領域の並び，… ；
do：出力共通領域の並び，…)

図12 インタフェース情報の表現法

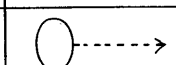
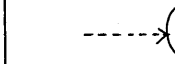
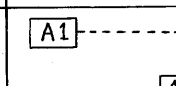
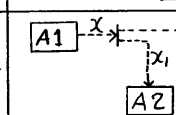
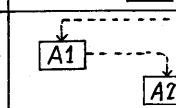
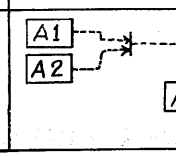
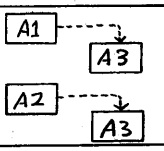
図12において，各インタフェース並びの命名はアローのラベルを用いる．

次に，略図から言語表現への変換基準を示す．

(i) インタフェース情報については略図から，ほぼ機械的に定まる．

(ii) 呼び出し制御のアルゴリズムは表5の規則で変換する．

表5 略図から言語表現への変換

1		entry
2		return
3		begin A1 A2 または end A1 goto L1 ⋮ L1:A2
4		case of x x1 : A2 x2 : A3 end
5		loop A1 A2 end
6	 は 	とみなして，3に従う．

(3) (1)，(2)の手順を各図式ごとに繰返すことでモジュール階層構造が定まる．しかし，各モジュールのコーディング量が著しく少ない時は，呼び出しに伴なうオーバーヘッドを考慮して，以下の範囲内で手直しをする．すなわち，同一レベルのモジュールどうしの併合，もしくは直接の親モジュールへの吸収の2種類にかぎって行なう．

4. おわりに

本稿はプログラムの機能表現から構造表現への自然な変換活動を支援する設計法，作業支援図式について報告した．

ただし，機能設計で同一階層と認識された場合でも，モジュール階層構造に変換する場合に，同一階層内のボックスの間に呼び出し関係を定めた方がよい場合があるが，本稿では扱わなかった．

本設計法は，現在我々の研究室で使用と評価を開始しており，改良を重ねていきたい．また，本稿で述べた図式表現に対応する記述言語をもとにした設計支援ツールを開発していきたい．

本研究は，昭和56年度科学研究費補助金，奨励研究(A)56750237によった．

参考文献

[1] 落水浩一郎，"ソフトウェア仕様の定義，検査，修正支援の一方式"，情報処理学会第23回全国大会，1981

[2] 酒井，玉井，落水，"PDBにおける設計，改良支援ツールDS"，情報処理学会ソフトウェア工学研究会17-6，1981

[3] M.A. Jackson，鳥居宏次訳，"構造的プログラム設計の原理"，日本コンピュータ協会，1980

[4] D.T. Ross，"Structured Analysis (SA)：A Language for Communicating Ideas"，IEEE Trans. Softw. Eng.，VOL. SE-3，No.1，1977

[5] M.E. Dickover，C.L. McGowan and D.T. Ross，"Software Design using SADT"，Proc. of the ACM National Conference Seattle，1977

[6] G.J. Myers，久保，国友訳，"高信頼性ソフトウェア複合設計"，近代科学社，1976