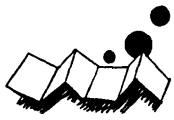
解　説

ソフトウェア開発に用いられる図形表現法†



青　山　義　彦††

1. はじめに

ソフトウェアの内容を表現する方法として，古くて新しいものが，「フローチャート」であろう．文献 1) によれば，「フローチャートは複雑な状態を目に見えるようにすることによって，混沌たる状態に秩序を与える助けとなるものである．フローチャートには，2つのレベルがあり，組織をつらぬくデータの流れを図式的に表現するトップレベルのものと（筆者注；分析，設計のため），所望の結果を得るためにとるべきすべてのステップを正しい順序で示す詳細レベル（筆者注；プログラムのため）のものとがある．」と説明している．現在では，フローチャートといえば後者の意味に一般に用いられているが，初期の段階ではフローチャートは正にソフトウェアの表現法の中心的役割をはたしていたといえる．

しかしソフトウェアが大規模化，複雑化するにつれて，ソフトウェア生産技法の研究開発が進む中で，フローチャートに代る新しい表現法が考案され，実用化されてきた．ソフトウェア開発の産物として，ドキュメンテーションの重要性が認識されはじめ，ソフトウェア内容の表現法への関心も高まってきた．すなわち（1）ソフトウェア全体を一度に分析，設計することは，一人の人間の能力を越え，どうしても中間過程を記録にとどめておく必要がある．また（2）チームによる開発においては，関与者間の良きコミュニケーションのために，あるいは保守のために，さらには（3）表現法は，分析，設計という人間の知的活動そのものと深くかかわりをもっているので，表現法のいかんは，分析，設計効率に大きな影響を与える等の理由から大変重要な要素を占めるものである．

本稿では，ソフトウェア要求分析フェーズ以降を対象に，過去開発されてきた技法の中から，特に図形表現法を備えた主要なものを概観する．もちろんドキュメント量の面からみると図形表現の占める割合は多いとはいえないし，また図形表現が適切か否かについても意見のあるところであるがビジュアル性等の長所もある．そこで図形表現法か，テキスト表現法かの問題についての研究についても触れることにする．

2. ソフトウェア要求分析技法と図形表現法

1977 年頃までのソフトウェア要求分析技法の動向については，文献 2) に詳しく紹介されているので，本章ではそれ以後の主な技法を図形表現法の観点から紹介する．

（1） データフローダイアグラム (Data Flow Diagram)

まずとりあげなければならない技法に「データフローダイアグラム」がある．DFD には，1979 年頃，期を同じくして発表された Gane と Sarson によるものと[3]，DeMarco[4] によるものとがある（図-1）．図形シンボルは両者異なるが，いずれもデータの流れを中心に分析する点では同じ考え方によるものである．基本的には，4 種のシンボルを用いて，現行システム (physical DFD) あるいは論理システム (logical DFD) を記述し，段階的に詳細化し，コンピュータ化すべき範囲（ソフトウェアで実現する）を切り出すという分析手順を与え，それにもとづき「データ構造」「プロセスロジック」及び「ファイル構造」を分析し，その結果をソフトウェア要求仕様書としてまとめ以後の設計へ引きつぐ手順をとっている．

この方法は，過去我々が経験的にとっていたアプローチを実務の立場から体系化したものということができ馴じみ易い技法の 1 つであると考える．と同時にソフトウェア設計（後述する Yourdon and Constantine の DFD 設計法）へつながっていく一貫性をもっているので実用性が高いように考える．

（2） SAMM (Systematic Activity Modeling Method)[5]

Stephens と Tripp によるもので，「木構造 (tree

† Diagrammatical Method for Software Development by Yoshihiko AOYAMA (Systems Development Laboratory, Hitachi Ltd.).

†† (株)日立製作所システム開発研究所

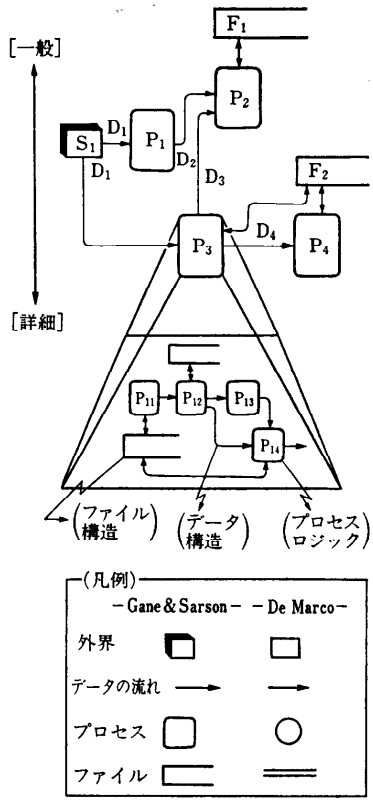
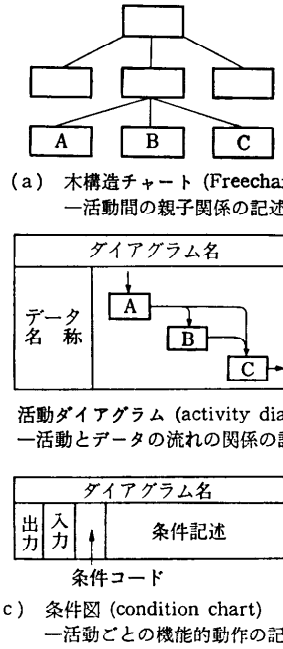


図-1 Gane Sarson[3] と DeMarco[4] のデータフローダイアグラム



(a) 木構造チャート (Freechart)
—活動間の親子関係の記述—

(b) 活動ダイアグラム (activity diagram)
—活動とデータの流れの関係の記述—

(c) 条件図 (condition chart)
—活動ごとの機能的動作の記述—

図-2 SAMM[9] の表現体系

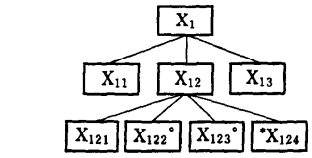
structure)」「活動ダイアグラム (activity diagram)」と「条件図 (condition chart)」の3つの要素からなる表現体系を用意しており (図-2), コンピュータツール化もされている. また有向グラフと結びつけることによって仕様の検証能力をもたせた点に特長がある. 中尾他による「FRAME」[6], 平他による「C-NAP」[7] もこれと同種の技法といえる.

(3) CORE (Controlled Requirement Expression)[8]

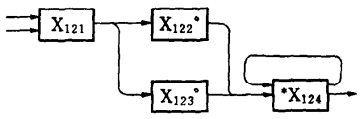
Mullery の研究によるもので時間の流れと時間的に順序づけられた項目間の前後関係を示す「動的ダイアグラム」と動的ダイアグラムをそれぞれ関係づける階層構造を示す「静的ダイアグラム」の2種類のダイアグラム (箱で表現) を用意し, それぞれの記述法を与えている (図-3). 記述法の中で相互排反関係 (◻ で表現), あるいは繰り返し (◻ で表現) の表現はジャクソンによるデータストラクチャチャートの表現法が導入されている. また本ダイアグラムは自動化支援ツールで有名な PSL/PSA[9] と接続されている.

(4) IA (Information Analysis)[10]

ストックフォルム大学の Longefors と Lundberg によって 1979 年頃に開発された技法で, 5つのステップからなる分析・設計手順と図形表現法を用意している. すなわち (a) 現状業務の記述及び業務モデルの記述を行う変更分析 (change analysis) と (b) 開発すべき情報システムの記述を行うアクティビティスタディ (activity study) に用いられる「アクティビティグラフ (A-グラフ) (c) 情報の先行関係の記述のための「情報フローグラフ (I-グラフ)」と情報を構成する要素の記述のための「コンポーネントグラフ (C-グラフ)」による情報分析 (information analysis) 過程, そして第4番目のステップとして, (d) データの全体構造の記述に「データ・システム設計グラフ (D-グラフ)」, 入出力データ構造の記述に「データ構造グラフ (D-ストラクチャ)」, データ構造にもとづくプログラム構造の記述のための「プログラム構造グラフ (P-ストラクチャ)」を用いて行われるデータシステム



(a) 静的ダイアグラム (static diagram)



(b) 動的ダイアグラム (dynamic diagram)

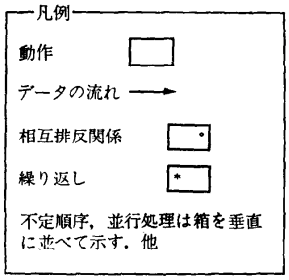


図-3 CORE[8] のダイアグラム表現法

設計 (data system design) の過程を経て，最後に，(e) D-グラフ上に具体的機器を割付けた結果を記述する「機器割付済データシステムグラフ (E-グラフ)」を作成し設計が完了する手順を与えている．(d) のデータシステム設計における「D-ストラクチャ」及び「P-ストラクチャ」の表現法はジャクソン法と同一である．全体的に図形表現法に関しては，必ずしも精練されているとはいいがたい面があるがデータを中心とした分析設計法の初期段階の代表的技法の1つといえる．すべての図形表現例は紙面の関係から文献にゆずるとして IA の全体構成と「I-グラフ」の例を 図-4 に示しておく．

(5) フローネット技法[11]

西尾他の研究によるもので，対象システムを非同期的に動作し続けるプロセスの集合としてとらえ，プロセス間通信，プロセスの動作を厳密に，かつ分り易く表現することを目的にしている．プロセス間通信を表現するための「システムフロー」と各プロセスの機能的動作を表現するための「データフロー」による表現体系をもっている (図-5)．データフロー表現にはペトリネットの動作規則を拡張導入している．ペトリネットの考え方を基礎とするものには，その他にも多くみられるがたとえば，Balkovich and Engleberg (GRC)[12] による研究もその1つである．

(6) その他の技法

以上紹介したもの以外にも図形表現法を備えた技法は多く存在する[13),14)]．Ross による「SADT」[15),16)]，Alford による「SREM」[17)~19)] はその代表であるが，すでに多くの紹介がなされているので割愛する．また Teichroew による自動化支援ツール「PSL/PSA」[9] も同様割愛した．SADT を支援ツールに関する研究

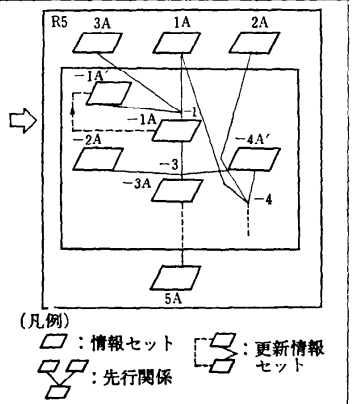
フェーズ		内容	グラフ名称	情報フローグラフ (I-グラフ) の例
(a)	変更分析	現行業務の記述 業務モデルの記述	アクティビティグラフ (A-グラフ)	
(b)	アクティビティスタデイ	開発すべき情報システムの記述		
(c)	情報分析	情報の先行関係の記述	情報フローグラフ (I-グラフ)	
		情報を構成する要素の記述	コンポーネントグラフ (C-グラフ)	
(d)	データ・システム設計	データシステムの全体構造の記述 (プログラム境界)	データシステム設計グラフ(D-グラフ)	
		プログラム入出力データ構造の記述	プログラム指向データ構造グラフ (D-ストラクチャ)	
		データ構造に基づくプログラム構造の記述	プログラムコントロール構造グラフ (P-ストラクチャ)	
(e)	機器割付	D-グラフに機器を割付けた結果の記述	機器割付済データシステムグラフ (E-グラフ)	

図-4 IA 技法[9]の概要と情報フローグラフ例

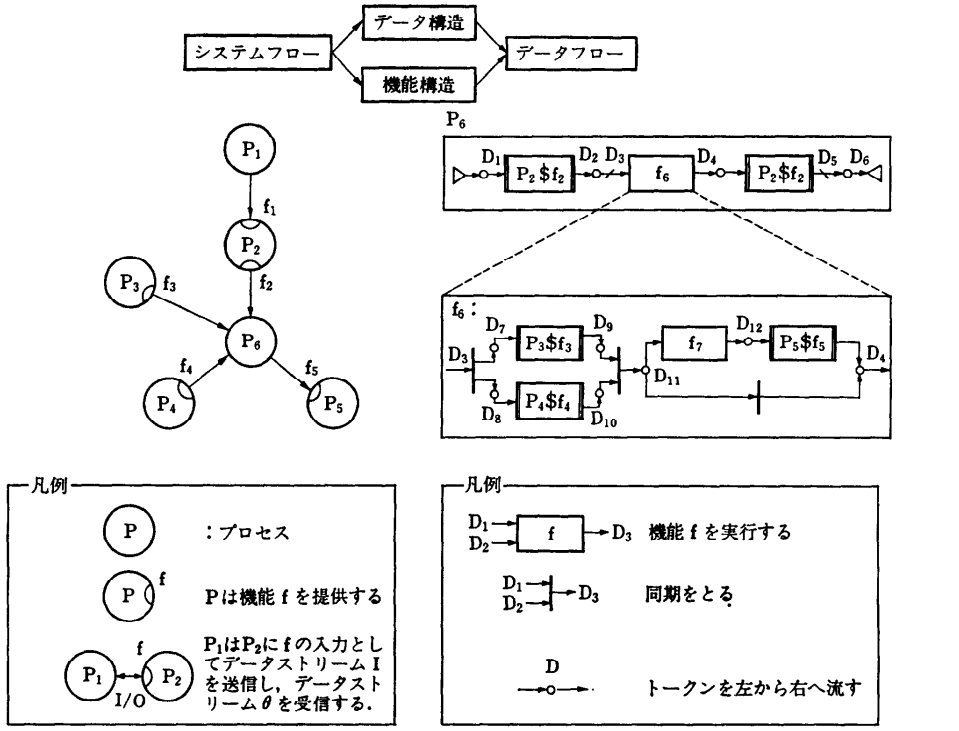
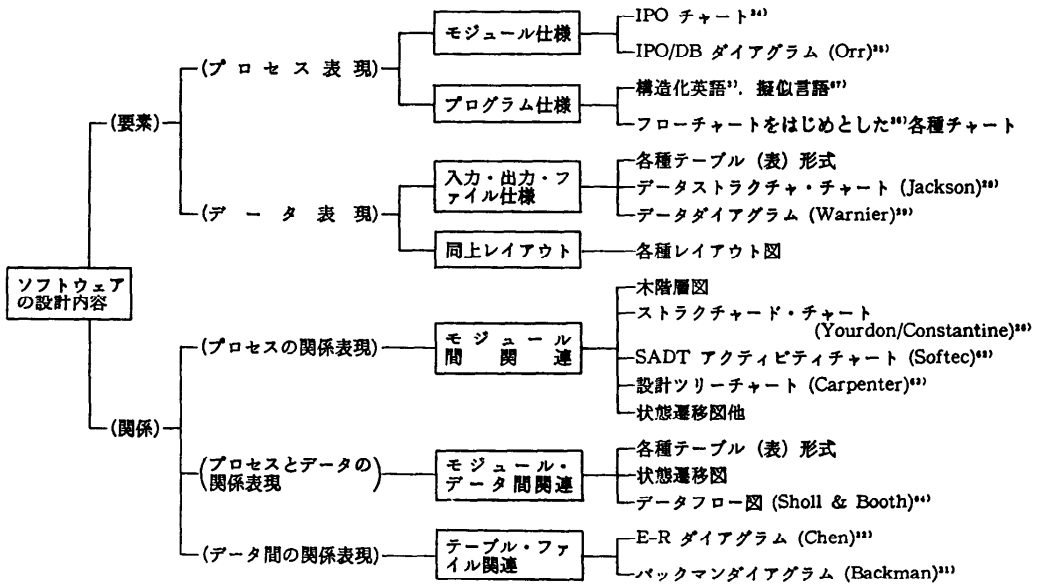


図-5 フローネット技法[10]による表現法

表-1 ソフトウェア設計ドキュメント体系と表現法



としては，小島他による「SADT サポートツール DST」[20,21] がある．

3. ソフトウェア設計技法と図形表現法

ソフトウェア設計は，ソフトウェア要求仕様（What to make)にもとづきソフトウェアとして実現すべき機能の分割（How to make）を行い「モジュール仕様」「モジュールインタフェース仕様」さらには「データ構造」の設計を中心とする基本設計と，手続（ロジック）を詳細化する詳細設計の2段階を踏んでソフトウェア設計書をまとめる過程といってよい．この過程で作り出される基本ドキュメントの体系とそれに用いられる表現法の主要なものを，**表-1** に示した．

ソフトウェア設計技法については，文献 22)～25) に詳しい解説，比較がなされているので，それにゆずることとし，表-1 に従って図形表現法についての簡単な紹介をする．

(1) プロセスの論理関係の図形表現法

一般に「モジュール関連図」「機能階層図」と呼ばれるものが基本設計過程で作られる．その設計結果を表現する方法として，静的な関係表現としては，「木階層図（tree chart）が，動的な関係を表現するものとしては，Yourdon and Constantine による「ストラクチャード・チャート」[26] が有名である（**図-6**）．ストラクチャード・チャートは，DFD から導かれ，モジュール間の親子関係，モジュール間の制御の流れ及びデータの流れを統合的に表現したものといえる．Willis and Jensen によって，ストラクチャード・チャートのツール化がなされている[27]．

(2) データの論理関係の図形表現法

入力データ構造，出力データ構造を表現するものの代表としては Jackson による「データストラクチャ・チャート」[28] と Warnier の「データダイアグラム」[29] がある．両者ともデータオリエンテッド設計法でよく知られており，データ構造を定義することからプログラム構造を導き出すことを主眼としており，そのためのデータの論理構造の表現法を与えているものである（**図-7**）．

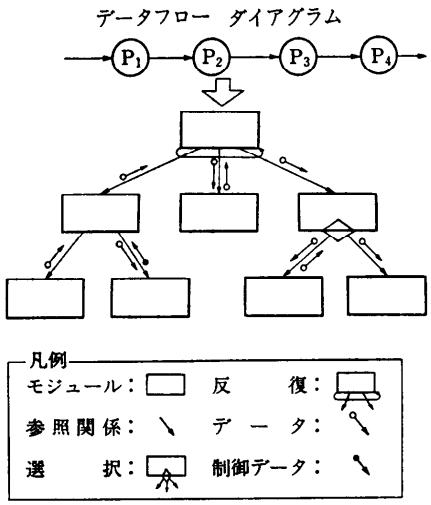


図-6 ストラクチャード・チャートの例[26]
(Yourdon/Constantine)

(3) データベース論理構造の図形表現法

データベース設計に用いられる新しい技法の研究が注目されている．これに関しては，文献 30) に詳しく解説されている．データベース設計はデータ間の関係をデータベースマネジメントシステムに独立な概念スキーマとして設計する「概念設計」と，データベー

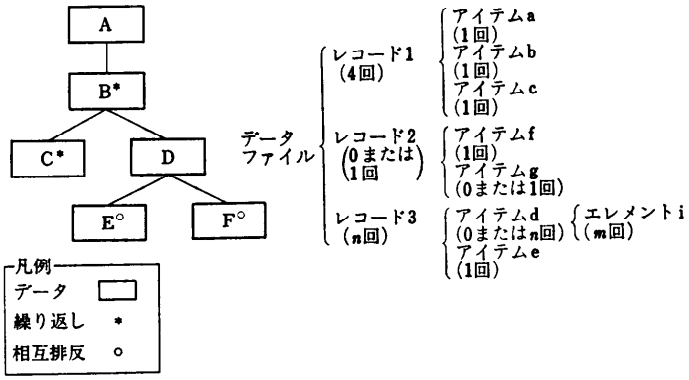


図-7 データ構造の表現法[28),29)]
(Jackson＝データストラクチャ・チャート(左)
Warnier＝データダイアグラム (右))

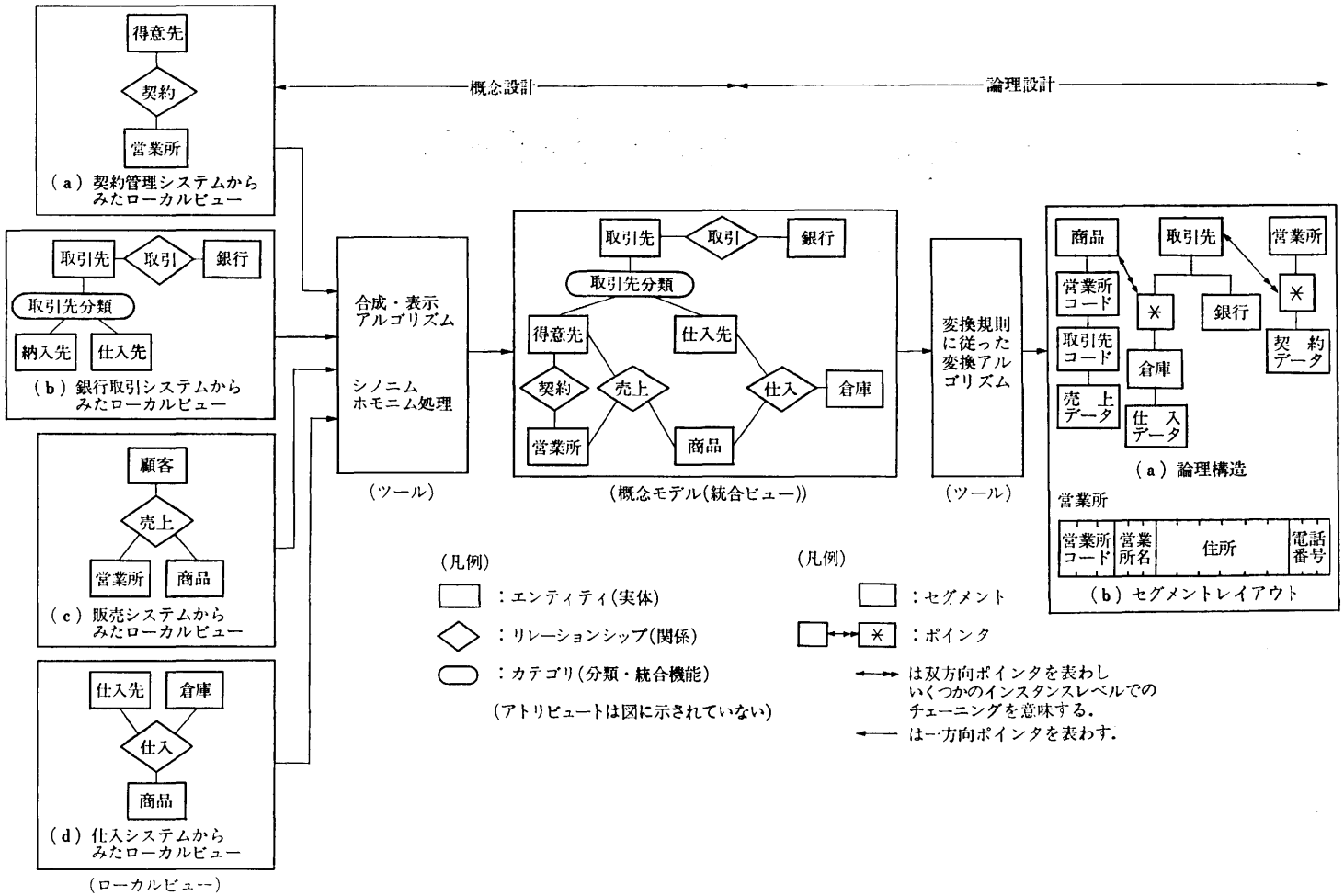


図-8 データベース設計法と図形表現例[33] (Chen による ER ダイアグラムを拡張している)

スマネジメントシステム（階層型，ネットワーク型，リレーショナル型）に従属する論理スキーマを設計する「論理設計」そして最終的にディスクファイル上の最適レイアウトを決定する「物理設計」の3つの過程を経て進める考え方がとられつつある．論理設計に用いられる図形表現法としてのバックマンダイグラム[31]は初期のものであるが，近年概念設計に Chen の ER ダイアグラム[32]を用いた研究が進んでいる．

図-8 に示した例は，近藤他[33]によるもので，Chen の ER 記法をベースにデータアブストラクション記法，及びインスタンス記法に含まれるいくつかの重要な要素を加え拡張したもので，概念設計から論理設計への手順とそのツール化を試みている．

（4） 機能の外部・内部仕様の図形表現法

機能の外部仕様，いわゆる「モジュール仕様書」は入力機能，出力機能，及び処理の3要素を記述したものであるが，「IPO チャート」[34]は代表的表現法の1つである．IPO チャートを拡張したものとして，Orr による「IPO/DB ダイアグラム」[35]がある（図-9）．

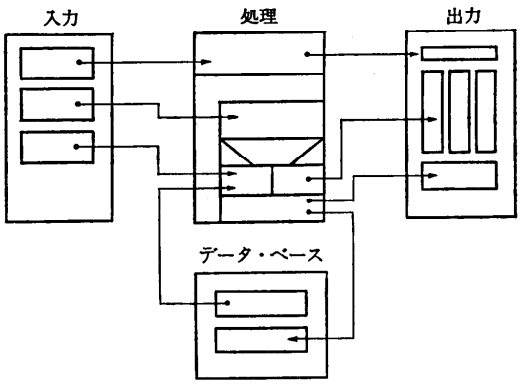


図-9 IPO/DB ダイアグラム[35]

（5） プログラムの内部論理の図形表現法

いわゆる従来フローチャートで記述されていたものであるが，フローチャート批判以後，構造化プログラミング法を踏まえた各種の図形表現法が開発され，実用化されてきている．この分野については，文献36)に解説されているので，その体系を示すにとどめた（図-10）．これ等のコンピュータツール化に関する研究開発も大変多くみられるようになってきている[37)-40)]．

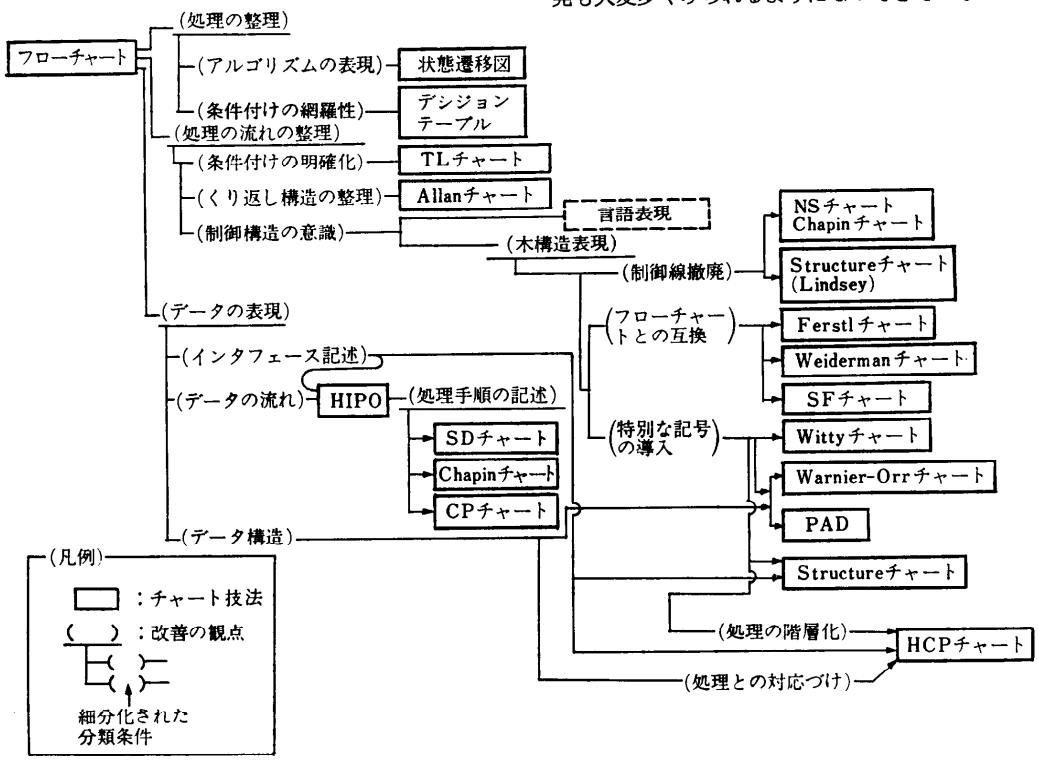


図-10 プログラムの内部論理の表現法
(文献 36) 図-4 "プログラミング用チャートの体系" p. 385 より著者の了解を得て転載)

4. ソフトウェアテスト技法と図形表現法[41)~43)]

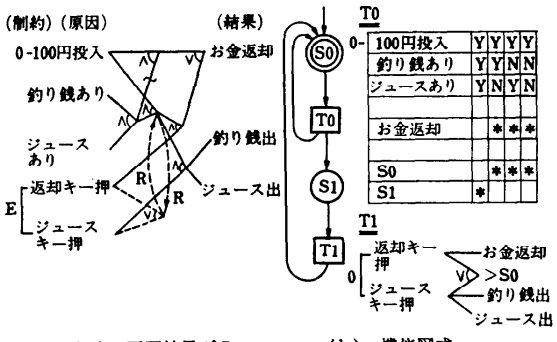
エラーのない完全に正しいプログラムを作ることは不可能で，限られた時間の中で可能な限り正しく動くプログラムを作るために，(1)機能テストのための「テストケース」のはたす役割は大きい．このテストケースの作成には，「原因結果グラフ」[42)]あるいは「状態遷移図」[44)]が代表的な技法として用いられている．古川他は，この両者の技法を統合した「機能図式」[45)]による方法と，そのツール化を進め実用化している(図-11)．(2)一方プログラムの内部の論理構造のテストを目的とするものが，「構造テスト」であるが，このテストでは，プログラムの内部構造を有向グラフ（制御フローグラフ）で表現しそれをもとにテストケースを作成する．しかし大変な数のテストケースになる場合が多く，実務上は先の機能テスト実施後に対象プログラムのカバレージを測定し，決められた値に達しない部分をまず つかみ，それを満足させるようなテストケースを作成してテストを行いカバレージ率を向上させていく方法がとられる．ツール化することによって構造テスト作業効率を向上させることが可能で既に実用段階に入っており，ソフトウェアの品質向上に貢献している[46)]．

5.「図形表現法」か「テキスト表現法」か

記述法として図形がすぐれているか，テキストがすぐれているかの問題にこたえることは容易ではない．

ここでは，この分野の研究について簡単に紹介する．

(1) R. Ramsey 他が[48)] PDL (Programming Design Language) とフローチャートについて実験的に比較評価した結果，設計段階では設計の全体的評価，設計時間等からみて，PDL の方がフローチャートよりも有用であるが，反面プログラミング段階では，設計書理解テストの成績や理解に要した時間等からみる限りにおいては，PDL とフローチャートの間に有意な差は検出されなかったことを報告している．



(a) 原因結果グラフ　　(b) 機能図式

図-11 原因結果グラフと機能図式[45)]（状態遷移図＋原因結果グラフ＋デシジョンテーブル）

(2) また，GE の S. Sheppard[49)] 他の研究によると，仕様書書式の特質を「記号」(自然語，制約語＝PDL，イデオグラム＝フローチャート) と「空間配置」(順次型，分岐型，階層型) としてとらえて，実験計画法を適用した結果，①記号の型は仕様書の理解に影響を与える．(理解状況を調べた質問に対し，自然言語記述は制約言語及びイデオグラムに比較して，有意に長い解答時間を要した結果が出た) ②空間配置の型は仕様書の理解に影響を与えはするけれども，記号の型ほどの大きな差はなかった．(仕様書の理解に要する時間が分岐型が最も短かく順次型が最も長いという傾向がうかがえるが，階層型については，問題によって異なる) という興味ある結果を報告している．

(3) 小滝他の研究[50)]では，「PDL が書き易く，PAD が読み易い」という仮定をおいて，両者を統合したツールである「PDL/PAD」[38)]の開発を進める中で，その仮定の正しさを実験を通して検証を試みている．それによれば「読み易さ」を「論理の追い易さ」と考えて，プログラムの複雑性は条件文の分岐の入れ子の数としてとらえると「条件文の分岐数の入れ子の数が少ない時 (1～2) では，PAD と PDL との読み易さには差はあらわれなかったが分岐数が3以上程度から差が出はじめた」という結果を報告している．

(4) 記述法の使い易さの側面として ①習得のし易さ ②読み易さ(出力のわかり易さ) ③問題解決の手助けとなる ④書き易さ (入力のし易さ) ⑤表現力の高さの5つの側面をあげることができるのではなかろうか．表現法としてはすべての側面を満足することが望ましいが，各側面は，読み易いものは書きにくい，あるいは表現力は高いが習得するのがむずかしいというように相反する関係にあり一意的に決めかねる要因から成り立っている．図形表現法は，読み易いと一般に考えられているが，その反面書きにくい，書く手間がかかる面がある．また表現能力に限界があり直観に訴える力が強いだけに，解釈に主観性が混入し

易く間違いを起こす面もある．したがって両者の特質を踏まえた複合型ドキュメンテーション体系を，それぞれの立場で確立していくことが望まれるわけであるが，そのための基礎的な研究が必要であろう．

この分野の研究はまだ未熟な点が多いことは否めない．ソフトウェアが人間の知的活動であるとするならば，人間的側面からの研究の進歩が今後期待されるところである．

6. おわりに

ソフトウェアの図形表現法について，文献を通して知りうるものについて紹介した．もちろんすべての分野をカバーしたとはいえない．たとえば近年マイクロコンピュータの広範囲な適用に伴い，シーケンス制御分野において用いられてきた「ラダー図」に代わる新しい記述法（表現法）の研究等がそれである．ステップダイヤグラム（LEGRFCET)[51]，長谷川によるマークフローグラフ[52]，薦田他による C-net[53]，加藤他によるシーケンスフローグラフ[54]等多くの提案がなされている．

1968年「ソフトウェアの危機」がいわれて以来多くの技法が，それぞれの問題意識をもって開発され，多種多様な図形表現法が生まれてきた．今後研究が進むにつれ過去の技法の拡張，改良が行われると共に新しい図形表現法が登場してくるであろうがそれと平行して，適用分野の観点から個々の表現法を体系化する努力が，実務の立場からは必要であると考える．ドキュメンテーション技術は，ソフトウェア生産技術の基礎をなすものであるだけに，ライフサイクルを通しての一貫性をもった体系化を，それぞれの分野で，実験一評価の上に積み重ねていくことが必要な時期にきていると考えるからである．

つぎに，図形表現法が実用化され，定着するためにさけて通れない課題の1つに，汎用性をもった図形処理端末（ワークベンチ）とツール化がある．近年ハードウェア技術の進歩により実現可能性が出てきており研究開発も盛んになってきた[55)-58)]．ツールの具備する条件として，①会話性（Interactive）②一貫性(Integrate) ③使い易さ（User friendliness）④可搬性(portability) の4つが一般にいわれているが，さらには，ソフトウェア開発の全体の環境(Software Environment) の中で考えようとする傾向が強く出てきた点も注目される[59),60)]．

ソフトウェアの生産性，信頼性，保守性，管理性の向上の観点から図形表現法がはたす役割，効果性についての研究はまだ少なく，確かなことがいえるまでにはいたっていないけれども，図形表現法は，ツール化技術の進歩とあいまって，実務の場では今後ますます普及していくことは予測に難くないところである．したがって図形表現法の観点から，個々の表現法をドキュメンテーション全体の中に位置づけて体系化することも意味のあることと考える．

参考文献

1) McCraken, D. D. et al.: Programming Business Computers, John Wiley & Sons, Inc. (1959)（高橋 茂監訳：事務用計算機のプログラミング(上)，p. 351，光琳書院，pp. 29-48(昭35)．
2) Ramamoorthy, C. V. and So, H. H.: Software Requirements and Specifications: Status and Perspectives, Tutorial: Software Methodology, IEEE (1978).（国井編：ソフトウェアツール，bit 2，共立出版，pp. 63-119 (1982) に邦訳されている)．
3) Gane, C. and Sarson, T.: Structured System Analysis/Tool and Technique, Prentice-Hall, p. 373 (1979)．
4) DeMarco, T.: Structured Analysis and System Spacification, Prentice-Hall, p. 352(1979)．
5) Stephens, S. A. and Tripp, L. L.: Requirements Expression and Verification Aid, proc. of 3rd ICSE (1978).（国井編：ソフトウェアツール，bit 2，共立出版，pp. 97-104 (1982) に邦訳されている)．
6) 中尾他：構造的アプローチによる情報システムの要求分析手法，システム工学会誌，Vol. 6, No. 1 (昭56)．
7) 平他：要求導出のための手順とツール：C-NAP，情報処理学会第25回全国大会論文集（昭和57年後期)．
8) Mullery, G. P.: CORE-A Method for Controlled Requirement Specification, proc. of 4th ICSE (1979).（国井編：ソフトウェアツール，bit 2，共立出版，pp. 87-96 (1982) に邦訳されている)．
9) Teichroew, D. and Hershey, E. A.: PSL/PSA: A Computer Aided Technique for Structured Documentation and Analysis of Information Processing Systems, IEEE Trans. on Software Engineering, Vol. 3, No. 1 (1977)．
10) Longefors, B. and Lundberg, M.: Information Analysis and It's Relation to Data Base Design and Programming Design, IAG International Seminar (1978)．
11) 西尾他：実行可能性を備えた要求定義モデル，情報処理学会第27回全国大会（昭和58年後期）

論文集.
12) Balkovich, E. and Engleberg, G.: Research Towards a Technology to Support the Specification of Data Processing System Performance Requirements, proc. of 2nd, ICSE (1976).
13) 国井監修：ソフトウェア工学-要求仕様技術, bit 8, 共立出版, pp. 137-144 (1978).
14) 野木：要求定義技術の動向, 情報処理, Vol. 20, No. 6 (June 1979).
15) Ross, D. T. and Schoman, K. E.: Structured Analysis for Requirements Definition, IEEE Trans, on Software Engineering Vol. SE-3, No. 1, pp. 216-230 (1977) (文献13) 中に邦訳されている).
16) Ross, D. T. and Brackett, J. W.: An Approach to Structured Analysis, Computer Decisions 8, 9, pp. 210-215 (文献13) 中に邦訳されている).
17) Alford, M. W.: A Requirements Engineering Methodology for Real-Time Processing Requirements, IEEE, Trans. on Software Engineering, Vol. SE-3, No. 1, pp. 180-193(1977) (文献13) 中に邦訳されている).
18) Alford. M. W.: Software Requirements Engineering Methodology (SREM) at the Age of Two; Compsac, Proceeding (1978).
19) Alford, M. W.: Software Requirements Engineering Methodology (SREM) at the Age of Four, Compsac, Proceeding (1980).
20) 小島他：SADT サポートツール DST, 情報処理学会第23回全国大会(昭和56年後期) 論文集.
21) 中村他：DST-C：データフローダイアグラムの検査システム, 情報処理学会第25回全国大会 (昭和57年後期) 論文集.
22) Bergland, G. D.: A Guided Tour of Program Design Methodologies, Computer, pp. 13-37 (Oct. 1981). (NIKKEI Computer 5-17, 5-31, 6-14 (1982) に邦訳されている).
23) Peters, L. J. and Tripp, L. L.: Comparing Software Design Methodologies, Datamation, (Nov. 1977).
24) Pressman, R. S.: Software Engineering; A Practioner's Approoch, McGraw-Hill, p. 352 (1982).
25) Parker, J.: A Comparison of Design Methodologies: ACM Sigsoft, Software Engineering Notes, Vol. 3, No. 4 (Oct. 1978).
26) Yourdon, E. and Constantine, L. L.: Structured Design: Fundamental of a Discipline of Computer Program and Systems Design, Prentice-Hall, p. 473 (1979).
27) Willis, R. P. and Jensen, E. P.: Computer Aided Design of Software Systems, Proc. of 4th ICSE (1979). (国井編：ソフトウェアツール, bit 2, 共立出版, pp. 157-166 (1982) に邦訳されている).
28) Jackson, M.: System Development, Prentice-Hall, p. 418 (1983).
29) Warnier, J. D.: Logical Construction of Systems, P Van Nostrand (1981).
30) 穂鷹：データベースの論理設計(1), (2), 情報処理, Vol. 24, No. 5, No. 7 (1983).
31) Bachman, C. W.: Data Structure Diagrams, SIGBDP 1, 2 pp, 4-13 (1969).
32) Chen, P.: The Entity-Relationship Model: Toward a Unified View of Data, ACM Trans. on Data Base Systems, 1.1, pp. 9-36 (1976).
33) 近藤他：データベースの論理設計技法, 日立評論, Vol. 64, No. 5 (May 1982).
34) 国友：効果的プログラム開発技法, 近代科学社, p. 276 (1979).
35) Orr, K.: Structured Systems Development, Yourdon Press, P. 170 (1977).
36) 佐藤：プログラミング用ドキュメンテーション, 情報処理, Vol. 22, No. 5 (May 1981).
37) 山崎：対話型図形作成支援ツール：DS, 情報処理学会第27回全国大会(昭和58年後期)論文集.
38) 前沢他：対話型の構造化プログラム設計製造支援システム(PDL/PAD)の開発, 情報処理学会第24回全国大会 (昭和57年前期) 論文集.
39) 髙野他：図形定義を用いたプログラム図生成, 情報処理学会第25回全国大会 (昭和57年後期) 論文集.
40) 米田：ソフトウェア設計のための図形スキーマシステム, 情報処理学会第25回全国大会 (昭和57年後期) 論文集.
41) 中所：ソフトウェアテスト技法, 情報処理, Vol. 24, No. 7, pp. 842-852 (1983).
42) 長尾 真監訳：ソフトウェアテスト技法 (G. J. Myers: The Art of Software Testing), 近代科学社, p. 192 (1980).
43) 日経エレクトロニクス：手工業的手法から脱皮を目指すソフトウェアテスト, No. 286, pp. 124-152 (1982).
44) Chow, T. S.: Testing Software Design Modeled by Finite-State Machine, IEEE, Trans. on Software Engineering, Vol. SE-4, No. 3 (1978).
45) 古川他：ソフトウェアテスト項目作成支援システム AGENT-II の開発と評価, ソフトウェア工学, 28-8, 2-9 (1983).
46) 鈴木他：系統的テストによる効果, 情報処理学会第25回全国大会 (昭和57年後期) 論文集.
47) 小滝他：ソフトウェア開発環境, 情報処理 Vol. 24, No. 6, pp. 707-714 (1983).
48) Ramsey, H. R. et al.: Flowcharts vs, Programming Design Languages: an Experimental Comparison, Proc. of the Human Factors Society, 22nd Annual Meetings (1978).

49) Sheppard, S. B. et al.: The Effects of Symbology and Spatial Arrangement on the Comprehension of Software Specifications, proc. of 5th ICSE (1981).
50) 小滝他：プログラム論理仕様記述法 PDL と PAD の「読みやすさ」における比較評価，ソフトウェア研究会，28-19 (Feb. 1983).
51) 戸塚：シーケンスの新しい表現法一仏 ADEPA の新提案の紹介(下)機械設計，第25巻，第2号 (Feb. 1981).
52) 長谷川：マークフローグラフに基づくプログラマブルコントローラ，電気学会，システム制御研究会，SC-83-17 (Feb. 1983).
53) 薦田他：FA用ステーションコントローラ(SCR)の提案，第9回 SICE システムシンポジュウム (Aug. 1983).
54) 加藤他：シーケンスフローグラフによるシーケンス制御システムの表現法，情報処理学会第25回全国大会（昭和57年後期）論文集.
55) 宇津宮：画面指向型 TSS 対話インタフェース HUMANICS-D の試作，情報処理学会第27回全国大会（昭和58年後期）論文集.
56) 杉崎：マルチウインドウを用いた利用者インタフェースの設計，情報処理学会第27回全国大会（昭和58年後期）論文集.
57) 山根他：ソフトウェアドキュメント作成方式の汎用化について，情報処理学会第27回全国大会（昭和58年後期）論文集.
58) 大川他：汎用プログラム図生成の実現方式，情報処理学会第27回全国大会（昭和58年後期）論文集.
59) Wasserman, A. I: Interactive Development Environments, Tutorial, Compsac Conf. (Nov. 1981).
60) Boehm, B. W. et al.: The TRW Software Productivity System proc, of 6th ICSE (1982).
61) Caine, S. H. and Gordon, K.: PDL-A Tool for Software Design. Proc. of NCC (1975).
62) Ross, D.: Structured Analysis (SA): A language for Communicating Ideas, IEEE Trans, on Software Engineering, Vol. SE-3, No. 1 (1977).
63) Carpenter, L. C. et al.: Systematic Development of Automated Engineering Systems, 2nd USA-Japan Symposium on Automated Engineering Systems (Aug. 1975).
64) Sholl, H. A. and Booth, T. L.: Software Performance Modeling Using Computation Structures, IEEE Trans. on Software Engineering, Vol. SE-1, No. 4 (Dec. 1975).

（昭和58年11月28日受付）