U.D.C. 681.32.06.002:658.512.2

ソフトウェア構造設計技法

Software Structure Specification Method

片岡雅憲* *Masanori Kataoka*
堂免信義* *Nobuyoshi Dômen*
野木兼六** *Kenroku Nogi*

大規模・複雑なソフトウェアの開発では，その構造の良否は当ソフトウェアの信頼性，生産性及び保守性に多大な影響を及ぼす。日立製作所では，汎用コンピュータ用ソフトウェアの構造の改善を目的として，昭和48年以来「ソフトウェア構造設計法」と呼ぶ標準技法を開発し，活用してきた。

当技法は，ソフトウェアの構造を論理構造と物理構造の両面からとらえ，各々についての系統的な設計指針を与えるものである。論理構造の設計では機能を論理的に厳密に定義し，機能を系統的に分割する。また，物理構造の設計ではプログラムを構成するモジュールやデータの階層構造，制御構造などを決定する。

当技法は，ソフトウェアの構造及び設計ドキュメントの標準化をもたらし，また，この標準化を前提とした設計支援治工具の実用化を可能とし，ソフトウェアの設計技術の改善に大きく寄与した。

1 緒　言

ソフトウェアの大規模化・複雑化につれ，その設計技術はますます重要なものとなってきている。ソフトウェアの設計では多様な要因を考慮に入れる必要があるが，中でもその構造の良否はソフトウェアの信頼性，生産性及び保守性を大きく左右する重要な設計要因である。日立製作所では汎用コンピュータ用ソフトウェアの構造の改善を目的として，昭和48年以来「ソフトウェア構造設計法」と呼ぶ標準設計技法を開発し，これを活用してきた。

本論文は当技法を紹介するものであり，ソフトウェア設計作業での当技法の位置づけを行ない，その内容を解説するとともに，当技法に基づいた設計ドキュメント及び設計支援治工具についても紹介する。

2 ソフトウェア構造設計法の位置づけ

2.1 ソフトウェア開発における設計作業の位置づけ

過去の時代では，ソフトウェアの開発とは，ややもすると「プログラムのコーディングを行なうことである。」ととらえられていた。

しかし，近年ソフトウェアの応用範囲が広がり，高度化・複雑化するにつれて，ソフトウェアの開発でも他の工業製品と同様に，設計，テスト，検査などの工程が必要とされ重要視されるようになってきた。このことは，ソフトウェアに対する認識が，単に「コンピュータを動かすための技術」とのとらえ方から，「高度で複雑な機能を実現するための抽象化機械」とのとらえ方に変わってきたことの証左とも言える。従来，手工業的製品であるとの批判を免れなかったソフトウェアを，近代工業製品に脱皮させるためには，このような新しいとらえ方に基づいた設計技術が必要である。

ソフトウェアの設計の目的は，ユーザー（市場，又は特定顧客）の要求に整合した，高い信頼性と保守性をもったソフトウェアを効率良く開発することにある。ユーザーから見たソフトウェアは，コンピュータを制御するプログラムと，その操作方法などを記述する外部仕様書（ユーザーマニュアルなど）から成る。ユーザー要求に整合したソフトウェアを実現するためには，**図1**に示すように，次に述べる三つの条件が必要となる。

(1) ユーザー要求に外部仕様（外部仕様書に記述されたユーザーから見た仕様）が整合していること。
(2) 外部仕様に欠落や矛盾がないこと。
(3) プログラムが外部仕様に整合していること。

ソフトウェアの設計工程は，**図1**に示すように基本設計，外部仕様設計及び内部仕様設計の3工程に分類される。基本設計ではユーザー要求を機能，性能，操作性，互換性などの面から分析して，開発のための各種計画書を作成する。外部仕様設計では，これらの計画書を基に外部仕様書を作成し，内部仕様設計ではこれを基に内部仕様書を作成する。

ソフトウェア構造設計法（以下，構造設計法と略す。）は，前記(2)及び(3)の観点から設計を改善するための技法である。すなわち，外部仕様に欠落や矛盾がないように，また，プログラムが外部仕様に良く整合するようにするための技法であり，外部仕様設計工程と内部仕様設計工程で用いられる。

2.2 論理構造と物理構造

ソフトウェアの設計で，その構造の設計は極めて重要である。ユーザーにとって理解しやすいソフトウェアは，その機能の構造が明確になっていなければならない。また，これを効率良く製造し保守するためには，プログラムを構成するモジュールやデータなどの構造が優れたものでなければならない。

ソフトウェアの構造は，**表1**に示すように分類される。同表の論理構造は，ソフトウェアの機能及び機能要素間の関係を規定する。一方，物理構造はプログラムを構成するモジュール，データ及びそれらの相互関係を規定し，具体的なプログラムの実現方式を指示するものである。したがって，**図1**で論理構造は外部仕様設計工程で決定され，物理構造は内部仕様設計工程で決定される。

構造設計法では始めに論理構造を，次に論理構造に基づいて物理構造を決定する。したがって，当技法で設計されたプログラムは，個々のモジュールがまとまった機能をもち，しかも相互間の独立性が高い優れた構造をもつことになる。

* 日立製作所ソフトウェア工場　** 日立製作所システム開発研究所

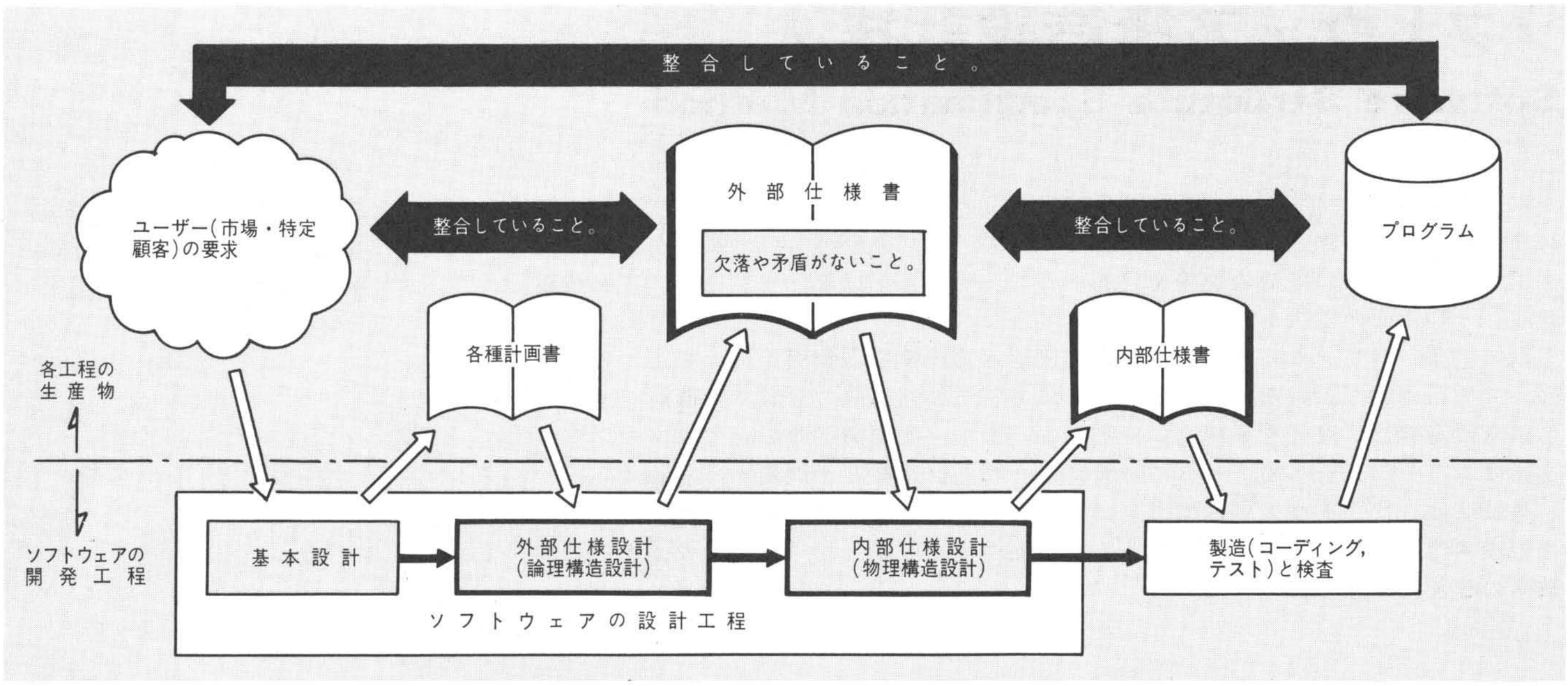


図1 ソフトウェアの開発工程の分類と各工程での生産物 ソフトウェアの開発工程は基本設計，外部仕様設計及び内部仕様設計の三つの工程に分類される。

表1 ソフトウェアの構造 ソフトウェアの構造は，論理構造と物理構造に分類できる。

分類	構成要素	要素間の関係
論理構造	機能	(1) 機能の階層 (2) 機能要素間の相互関係
物理構造	モジュール	(1) モジュールの階層 (2) モジュール間の制御関係
	データ	(1) データの階層 (2) データとモジュールの相互関係 (3) データ間のポインタ関係

3 論理構造の設計

論理構造の設計では，機能を入力と出力との論理的対応関係としてとらえ，また，機能要素間の論理的関係を決定する。

構造設計法は，論理構造を設計するための機能の論理的記述法及び機能の系統的分割法についての指針を与えている。

3.1 機能の論理的記述

本来，ソフトウェアの仕様は，明確な論理の上に組み立てられていなければならない。しかし，現実にはソフトウェアの仕様は日常言語で記述されることが多いため，必ずしも論理的に明確でないことがある。例えば，記述された仕様の論理を幾通りにも解釈できる場合がある。

このような不明確さを除くためには，その部分の入力と出力の関係を論理的に厳密に定義すればよい。構造設計法ではこれを表形式のデシジョンテーブル，又は図形式のＣＥＧ[1)]（Cause and Effect Graph：原因結果グラフ）によって行なう。**図2**は日常言語で記述された仕様をＣＥＧで表現した例であり，仕様の論理が二通りに解釈できることが分かる。このように日常言語で記述されていると論理的な不明確さが隠されてしまうが，ＣＥＧやデシジョンテーブルでは論理を厳密に定義して不明確さを取り除くことができる。

3.2 機能の分割

機械装置がなぜ動くかを知りたいとき，我々はその装置の仕掛けについて知ろうとする。ソフトウェアを設計するときも，仕掛けを考えて目的機能の実現を図ろうとする。構造設計法ではこの仕掛け，すなわちアルゴリズムを機能の系統的分割の手段としている。

図3に示すように，目的機能の実現アルゴリズムを考えることにより，目的機能は部分機能に分割される。ここで，分割は部分機能と部分機能間の関係をみれば，目的機能のアルゴリズムが読み取れるようなものでなくてはならない。重要なことは，第1段階の分割で部分機能の実現アルゴリズムを

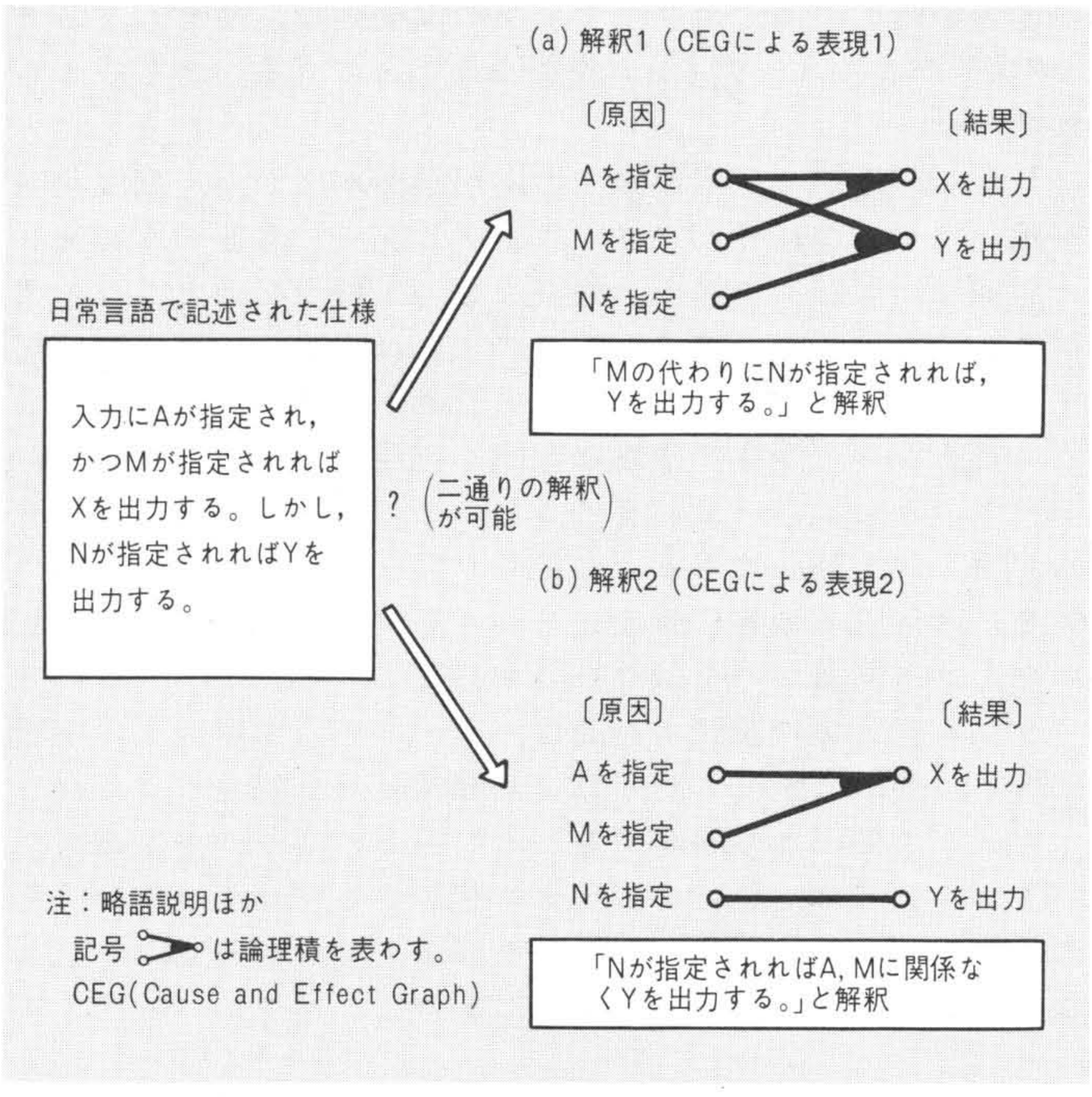


図2 日常言語で記述された仕様のCEG表現 日常言語で記述された仕様は，論理的な不明確さを含んでいる。

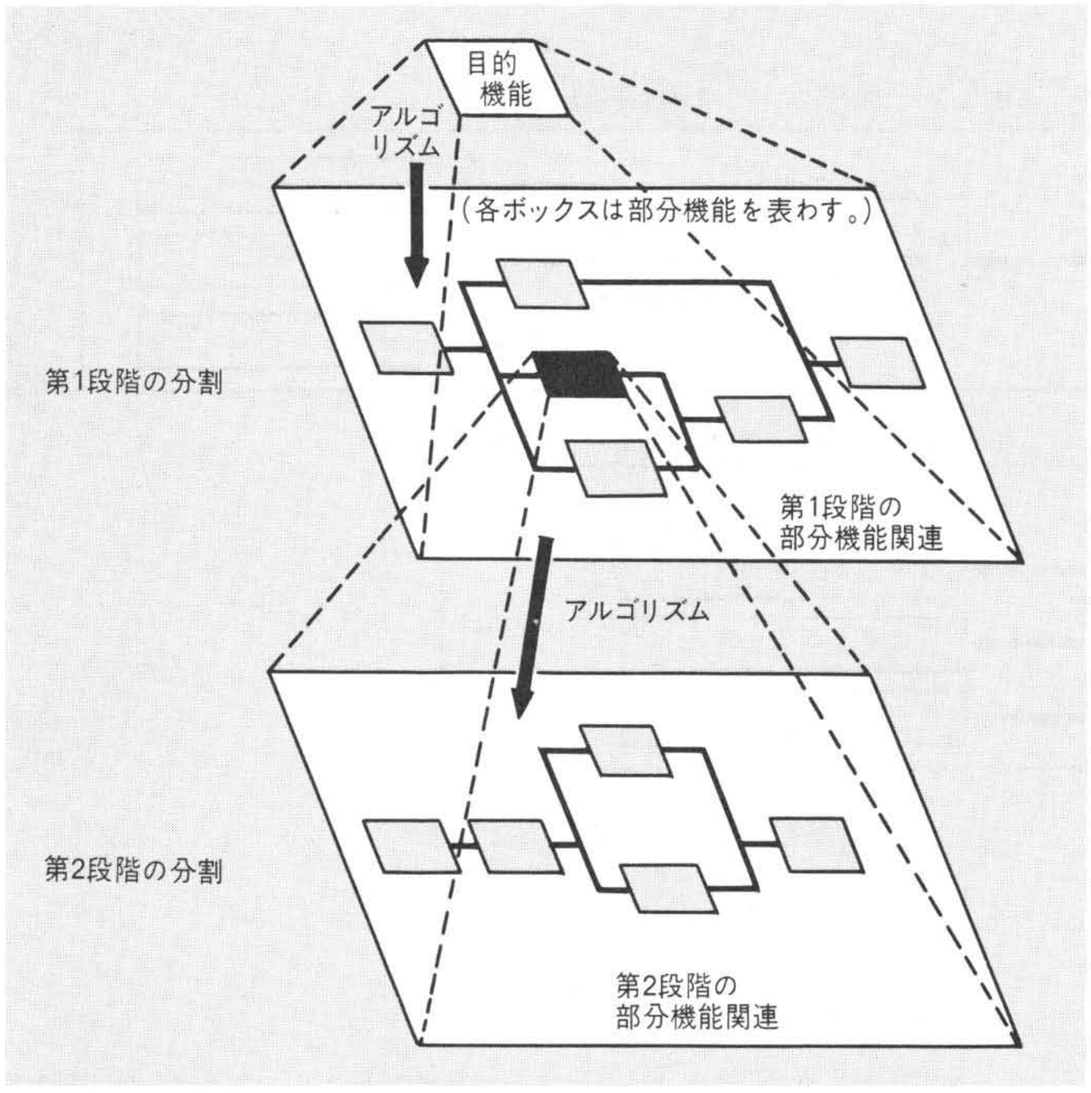


図3　機能の階層分割とアルゴリズム　機能はその実現アルゴリズムを考えることにより，部分機能に分割できる。すなわち，機能(又は部分機能)の実現アルゴリズムは，その一段下の関連図上に表現される。

記述してはいけない，ということである。部分機能の実現アルゴリズムは第2段階の分割で行なえばよく，第1段階で行なう必要はない。

次に，上記の機能分割で用いる基本的アルゴリズムを考えてみよう。ソフトウェアの機能は**図4**(a)に示すように，入力から出力への写像としてとらえることができる。したがって，機能の分割とは写像の分割であり，**図4**に示すように次の二通りの方法で分割できる。

(1)　入出力の分割による機能分割(横方向の分割)

図4(b)に示すように入力集合及び出力集合を分割することにより，機能 f を部分機能f_a，f_b，f_cに分割できる。例えば，事務処理用ソフトウェアで，入力ファイルや出力リストの種別により機能を分割することができる。

(2)　変換過程の分割による機能分割(縦方向の分割)

図4(c)に示すように入力から出力への変換過程を分割することにより，機能 f を部分機能g，h に分割できる。例えば，コンパイラでは入力(ソースライブラリ)を出力（オブジェクトライブラリ）に変換する過程を分割し，全体の機能をシンタックスチェック，中間語生成，コード生成，最適化などの部分機能に分割している。

以上述べたように，機能はその実現アルゴリズムを考えることにより部分機能に分割でき，また，これを繰り返すことにより次々に階層的に詳細化，具体化していくことができる。

4　物理構造の設計

物理構造とはプログラムの物理的構造，すなわちプログラムを構成するモジュールやデータの構造をいう。ここでいうデータはユーザーから見える入出力データだけでなく，プログラムの内部で用いられるワークファイルや制御表を含んでいる。構造設計法では物理構造を以下に述べる観点から系統的に設計する指針を与えている。

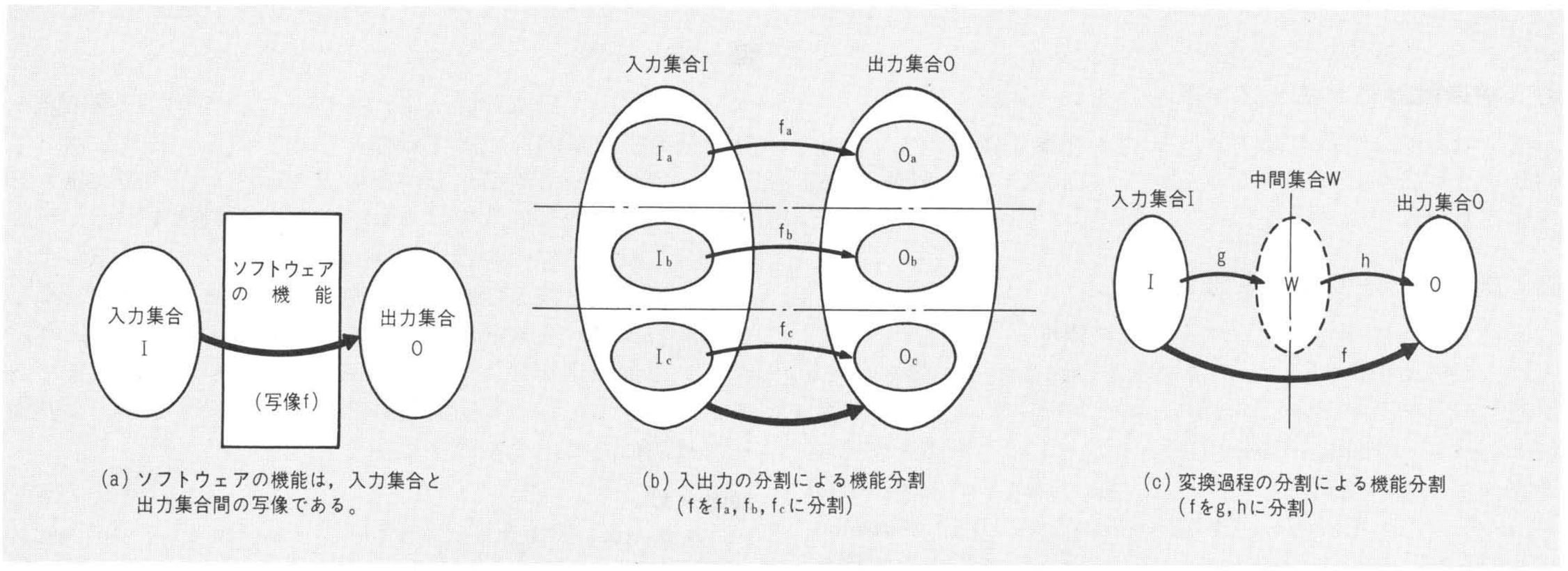
(1)　論理構造に整合した物理構造を実現する。
(2)　物理構造を標準化する。
(3)　物理構造上の構成要素の独立性を高める。

4.1　モジュール構造の設計

モジュール構造はモジュールの階層構造(静的な構造)とモジュール間の制御構造(動的な構造)とから成る。

構造設計法では，モジュールの階層構造をプログラム—デパートメント(10kステップ以下)—セクション(1kステップ以下)—ユニット(150ステップ以下)の階層となるように標準化している。モジュールの階層構造は **3.2**に述べた機能の階層構造に，共通化及び抽象という変換を加えることにより設計される。共通化では機能階層構造上の部分機能のうち共通の機能を抽出して，これを共通モジュールとする。また，抽象化では特定のデータを入出力とする部分機能群を集めて，これを一つの共通モジュールに格納する。抽象化によりこのデータは，共通モジュールの中に隠されることになり，他の部分機能は共通モジュールの機能だけを意識すればよいことになる。共通化及び抽象化はこのように機能を再整理する技法であり，プログラムの開発量を削減する技法と言える。

モジュールの機能間の関係を実現するために，モジュール

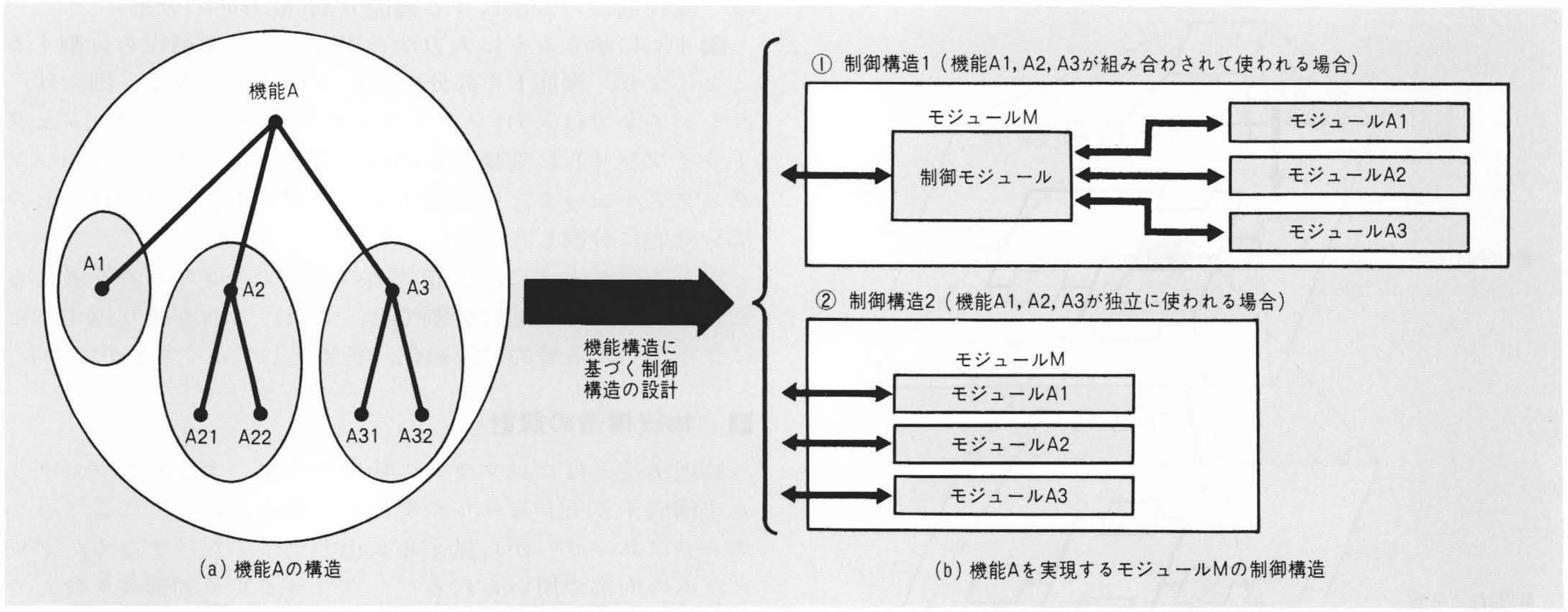


(a) ソフトウェアの機能は，入力集合と出力集合間の写像である。

(b) 入出力の分割による機能分割
(fをf_a, f_b, f_cに分割)

(c) 変換過程の分割による機能分割
(fをg, hに分割)

図4　ソフトウェアの機能分割の基本的アルゴリズム　ソフトウェアの機能は写像としてとらえることができ，写像の性質を利用して二つの基本的アルゴリズムで分割できる。



(a) 機能Aの構造　　(b) 機能Aを実現するモジュールMの制御構造

図5　モジュールの制御構造の決定　機能Aの構造に基づいて，これを実現するモジュールMの制御構造を決定する。

間の制御構造を決定する必要がある。構造設計法ではモジュールの独立性を推進する目的で，制御構造に関して次に述べる二つの基本的な規則を設けている。

(1)　モジュールは入口一つ，出口一つとする。

(2)　モジュールは閉サブルーチン形式（当モジュールをコールした次の命令に制御をもどす形式）とする。

上記の規則に従って，機能間の関係をモジュール間の制御構造に変換する例を**図5**に示す。

4.2　データ構造の設計

データ構造の設計はモジュール構造の設計と併行して行なわれ，データの階層構造，データとモジュールの相互関係及びデータ間のポインタ関係を決定する。

上記のうちデータの階層構造は，モジュールの階層構造に対応して設計される。例えば，**4.1**で述べたモジュールの階層構造上の特定のデパートメントで用いられる制御テーブルは，このデパートメントに属すテーブルとみなす。同様にしてセクションやユニットに属すテーブルが定義されて，データ全体としての階層構造が決定されると同時に，データとモジュールとの相互関係も決定される。また，データのポインタ関係は，このデータの処理性能などに影響を与える重要な要素であり，構造設計法ではこれについての設計指針を与えている。

5　構造設計用ドキュメント

構造設計法は，設計用ドキュメントを**表2**に示すように標準化している。これらのドキュメントは図表形式を原則とし，視覚化を図っている。

6　構造設計を支援する治工具

構造設計法はソフトウェアの構造の標準化を促進するとともに，設計用ドキュメントの標準化及び図表化を可能とした。そして，これらの標準化は更にソフトウェア設計のCAD（Computer Aided Design）化をもたらしつつある。構造設計を支援する治工具として，現在，AGENT[2]（Automated Generator of External Test-cases）とADDS[2]（Automated Design and Documentation System）の二つが実用化されている。AGENTはCEGの情報を入力することによりテスト項目を生成するテスト支援治工具として開発されたものであるが，テストだけでなく論理構造の設計を支援する治工具としても優れている。また，ADDSは物理構造の設計を支援する治工具であり，設計情報を解析し，図表形式の設計用ドキュメントを出力する。

表2　構造設計用ドキュメント　構造設計用ドキュメントは図表形式を原則とし，視覚化を図っている。

項番	ドキュメント名	形　式	目　　的
1	機能仕様書	図，表を含む文章	外部仕様を表わす（CEG，デシジョンテーブルを含む）。
2	機能階層構造	図	モジュールの階層構造を表わす。
3	モジュール仕様	表	モジュールの仕様を表わす。
4	データフロー	図	モジュールの入出力と処理手順を表わす。
5	モジュール関連	図	モジュール間の制御関係を表わす。
6	テーブル仕様	表	データの内容と利用方法を表わす。
7	テーブル関連	図	データ間のポインタ関係を表わす。
8	メモリマップ	図	モジュールやデータのメモリ上での配置を表わす。
9	解　説　編	自由	名称の付与方法，使用されるアルゴリズムの説明など項番1～8では表わせない設計上の規則を記述する。

7　結　　言

ソフトウェア構造設計法は，ソフトウェアの論理構造及び物理構造を系統的に設計する標準技法である。当技法はソフトウェアの構造の標準化と，設計ドキュメントの図表化を可能とした。また，最近ではこれを基にした設計支援治工具を実現し，設計作業の改善のための中核技術として成長した。

このように当技法は実用化技術としての軌道には乗ったものの，当技法が今後解決すべき課題は少なくない。例えば，データの抽象化などについての系統的技法の開発が必要である。今後，これらの技術課題の開発に鋭意取り組み，より優れた技法に育てていく考えである。

参考文献

1)　G. J. Myers，有澤訳：ソフトウェアの信頼性，p. 250～260，近代科学社（昭52-10）

2)　片岡，外：ソフトウェア開発支援システム“CASD”，日立評論，**62**，879～882（昭55-12）