コンピュータと教育 4－4
（１９８９．２．２５）

スタンドアロン型教育用ソフトウェアの設計方式

—— 開発経験からの事例報告 ——

平賀　正樹

富士通（株）

教育用ソフトウェアを設計する場合，その使用目的や教育（学習）効果などについて検討することが最も重要な課題である．しかし，教育や学習とは本質的に関係のない技術的な問題を避けては通ることは不可能であり，そのような技術的ノウハウの蓄積も不可欠であると考える．本稿では，ここ数年間にわたる，FMパソコン用のスタンドアロン型教育用ソフトウェアの開発経験による三つの事例：「構造化（設計）手法」により BASICで開発を行った例，「オブジェクト制御方式」によりアセンブリ言語で開発を行った例，そして，「階層メニュー構造プログラムの構築支援方式」により日本語プログラミング言語Mind で開発を行った例について報告する．

The method of designing "stand-alone" type educational software

— A case report from development experience —

Masaki Hiraga

Systems Section, Computer Assisted Learning System Dept.,
Systems Engineering Group, Fujitsu Ltd.
17-25, Shinkamata 1-Chome, Ota-ku, Tokyo 144, Japan

Although there have been many various arguments and reports discussing educational software, few refer to the technique of software development.

This paper reports on three examples of "stand-alone" type educational software:

i) development by "structured design techniques" using BASIC,
ii) development by "object control systems" using assembly language,
iii) development by "building assisted method of hierarchical menu structure programming" using the Japanese programming language "Mind".

This paper also describes the evaluation of these development methods.

1. まえがき

これまで教育用ソフトウェアの要件に関しては，いろいろ議論され報告もされてきているが，その開発手法にまで言及した例は，それほど多くはなかったように思われる．

本稿では，筆者らによる，これまでの（パソコン用）スタンドアロン型教育用ソフトウェアの開発経験の中からの三つの事例：「構造化（設計）手法」により BASIC で開発を行った例，「オブジェクト制御方式」によりアセンブリ言語で開発を行った例，そして，「階層メニュー構造プログラムの構築支援方式」により日本語プログラミング言語Mind で開発を行った例を報告すると共にこれらの開発手法の評価，および今後の展望等について述べる．

2. 構造化手法による BASICでの開発事例

2.1 構造化手法の概要

一般に，プログラム開発において構造化手法を用いる利点としては，信頼性の向上やプログラム（リスト）自体の読みやすさによる保守性の向上等を挙げることができる．しかしながら筆者らが構造化手法を導入した動機としては，複数のプログラマで開発することを考慮した点もあるが，当時社内事情により開発言語として BASIC（もしくはアセンブリ言語）しか使えない状況にあったことも一因している．

本件における教育用ソフトウェア開発では予め，下の①～④に示すような手順のプロトタイピング手法を用いることを考えた：

① 開発する教育用ソフトウェアの基本的な目的および必要な機能等の抽出，
② 構造化手法（ソフトウェアの論理構造を目で見て分かるような図形で表現することを基本思想としたもの）によるフローの作成，
③ （②が詳細レベルにまでおちていれば）コーディング，
④ 評価し，修正および機能強化項目を決めて②へ分岐．修正もしくは機能強化項目がなければ殆ど完成に近い状態．

2.2 教育用ソフトウェアの概要

ここで開発した教育用ソフトウェア[1]は，“ゲーム等のように無条件に面白く魅力的な状況が設定されていて，その魅力を損なわない範囲において教育的効果が加味されたものとする”という方針に基づいている．具体的には『かいものゲーム』（一種の思考型ゲーム）という設定にし，これは従来の一人で行うようなゲームではなく，二人で出来るという特徴がある［注：教育的効果は（かいもの行為による）計算力の向上，（ゲームの内容に起因する）確率の概念の理解等である］．なおこれは，相手との駆け引きの中に面白さがあるという考えに基づいている．

ただし，それまでの思考型ゲーム（通常のボードゲーム等も含まれる）によくありがちな，プレーヤー間に有利／不利の差が生じてしまうと（一発逆転のチャンスがあるものの）一般的に一方的なゲーム展開となり面白さに欠けてしまうという問題点を解決するため，「伯仲を演出する制御処理方式」を導入した．これは，ゲームの興味を持続させるために，ゲームを行っているプレイヤーの有利／不利な状況に追従する形でゲームが伯仲するようにダイナミックに制御を行わせるもので，次の①～③の機能を備えるように構成される：

① プレイヤーの有利者に対しては多くの障害を発生させ，不利者に対しては少ない障害を発生させるように障害発生確率を算出して制御する機能，
② 何れかのプレーヤーがゴールに到達する前に共倒れすることがないように，プレー中の得点を逐次計算する機能，
③ プレーヤー相互の得点差を算出し，無条件に与える得点の値および確率を制御する機能．

2.3 実施後の所感および評価

図1 は，構造化手法により設計した（フロー記述した）例である．これは最も初期レベルのものであり（コーディングは無理），一種のラフスケッチのようなものである．この状態から段階を経て詳細に入るわけである．

しかし実際に BASICによるコーディング（高速のグラフィック処理を要する部分等はアセンブリ言語で記述した）では，ソースプログラムは決して読みやすいといった状態にはならなかった．これは，下の①，②の理由による：

① 細かい仕様変更あるいはバグ修正等を重ねるうちにフローを書き替えずに，直にコーディングせざるを得ない状況があったこと，
② ハードウェアの制約により，肥大したソースプログラムに対し（手作業による）圧縮を施さなければならなかったこと．

結果として，保守性はフローの中に基本部のみ保たれていると言ってよいが，構造化手法については非常に有効であるとの実感を得た．

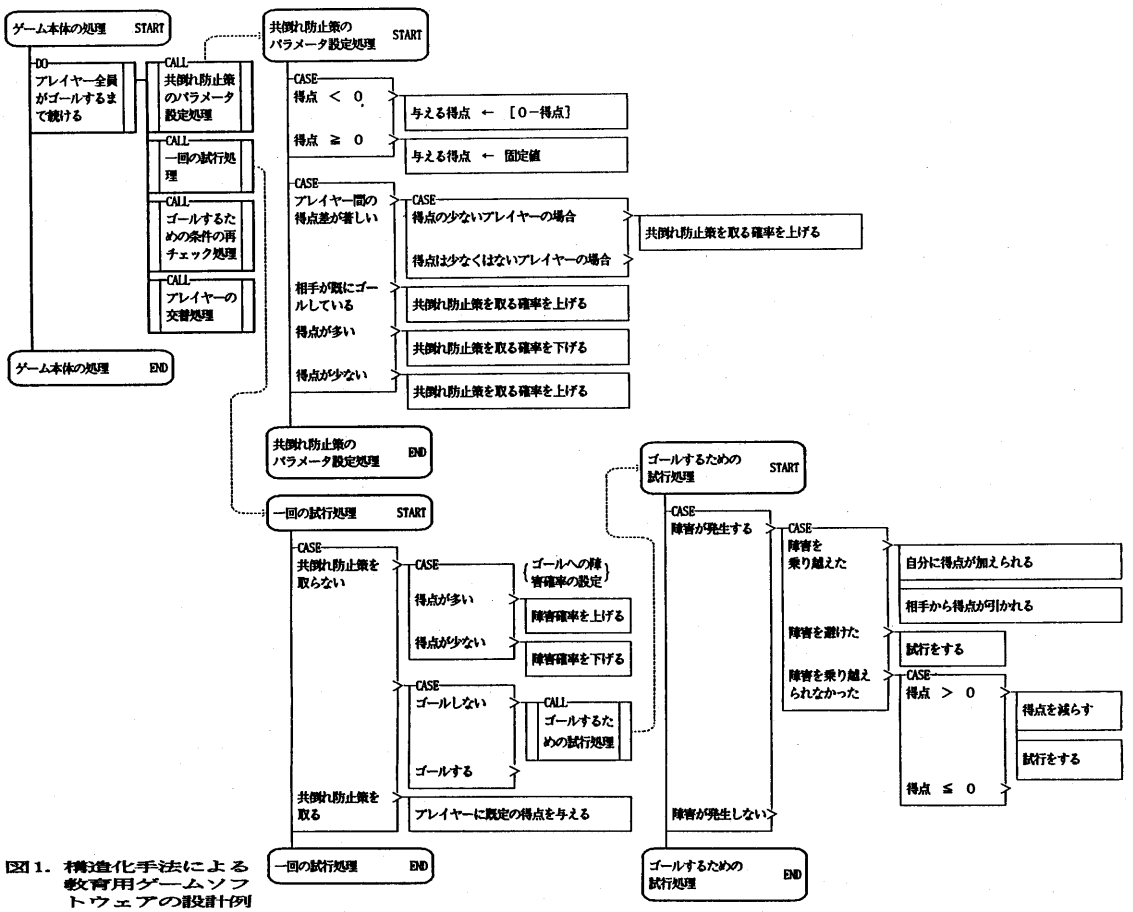


図1. 構造化手法による教育用ゲームソフトウェアの設計例

3. オブジェクト制御方式によるアセンブリ言語での開発事例

3.1 オブジェクト制御方式の概要

ここで述べるオブジェクト制御方式[2),3)]は，当時筆者らが抱えていた BASICを採用することによる速度性能や8ビット機 64KB というメモリ空間の制約という開発上の問題点を解決するために，データ加工・解析用対話型言語『SPEAKEASY 』[4)]や『OS-9』をヒントにして設計したものである．これは，8 ビットプロセッサ等を用いた小規模のデータ処理システムにおいて，単純な制御機構により，限られたメモリの効率的な使用を可能とすると共に，アセンブリ言語等で開発される応用システムの開発を容易化した動作環境を，提供することを目的としている．

図2 により，その基本構成について説明する．

この方式では，教育用ソフトウェア等の応用システムは操作単位となる複数のオブジェクトの集合として管理される．即ちシステム設計時に，そのシステムを構成するプログラムやデータを予め処理機能単位で分割した個々の要素がオブジェクトである（以下，プログラム・タイプ・オブジェクトを PTO，データ・タイプ・オブジェクトを DTOとする）．オブジェクトは，オブジェクト管理情報部とオブジェクト本体部とからなる．オブジェクト管理情報部は，サイズ情報，タイプ情報，レベル情報等のような情報を持つ．またオブジェクト本体部には，オブジェクトが PTOである場合にはマシン語コードが格納され，DTO である場合には数値や文字等のデータが格納される．

メイン制御部は，各オブジェクトの実行終了毎に，次オブジェクト名記憶部に設定されたオブジェクト名を取

り出し，共通ルーチン制御部を介して，共通ルーチン部におけるオブジェクト・サーチ処理部を呼び出す処理を実行する．そして，オブジェクト・サーチ処理部から通知されたオブジェクトのアドレス情報に基づいて，次のオブジェクトへ実行制御を渡す．

共通ルーチン部は汎用的に使用されるルーチン群からなり，各種ファイル入出力機能を提供するルーチン等が用意されている．これらの各共通ルーチンにはリクエスト番号が付与され，共通ルーチン・アドレス・テーブルに，そのリクエスト番号に対応する共通ルーチンのアドレス情報が記憶される．そして共通ルーチン制御部は，指定されたリクエスト番号によって，そのリクエスト番号が付与された共通ルーチンを呼び出す制御を行う．共通ルーチン中で重要なものとしてオブジェクト・サーチ処理部がある．オブジェクト・サーチ処理部は，指定されたオブジェクト名を持つオブジェクトが，オブジェクト用メモリ領域上に存在するか否かを，各オブジェクトのオブジェクト管理情報部を参照することにより検索する．該当するオブジェクトがある場合，そのオブジェクトの先頭アドレスを共通ルーチン制御部の呼び出し元へ通知する．ロードする時にオブジェクト用メモリ領域上に充分なフリー（空き）領域がない場合には，ガーベジ・コレクタを呼び出して不要となった領域を回収することによりフリー領域を確保する．

なお，メイン制御部による実行制御とは無関係に，PTO は，必要な DTOをいつでもロードすることができる．

次に，図3 (イ) に示す 8ビット機 64KB のアドレス空間を持つシステムのメモリにおいて，メモリのモニタ領域には，図2 に示す次オブジェクト名記憶部，メイン制御部，共通ルーチン制御部，共通ルーチン・アドレス・テーブル，共通ルーチン部が設定される．応用システム用の領域としては，モニタ領域の後に，オブジェクト用メモリ領域が設けられ，この場合では，オブジェクト用メモリ領域が，PTO 用の PTO領域と，DTO 用の DTO領域とに分けて管理されるようになっている．図3 (ロ) は，3 個の PTO：PTO-1，PTO-2，PTO-3 と 2個の DTO：DTO-1，DTO-2がロードされている状態を示している．PTO 領域と DTO領域は，各々先頭から順次使用され，残りの領域はフリー（空き）領域として管理される．図3 (ロ) の状態で，例えば PTO-3の実行が終了し，次に PTO-4が指定されたとすると，この時 PTO-4のサイズが調べられ，PTO 領域のフリー領域から PTO-4用のロード領域が確保できるかどうか検査される．フリー領域が小さいため確保できない場合には，図3 (ハ) に示すように，ガーベジ・コレクタにより不要となった PTO-2（レベル情報で決まる）が PTO領域から廃棄され詰め替えが行われる（PTO-4 は DTO領域に DTO-3をロードしている．

次に，PTO の制御移行例を図4 に従って説明する．例えば，図4 (イ) 図示の①～⑫の順番で，各オブジェクト PTO-1 ～PTO-7 間の制御移行がなされたとする．なお，各オブジェクト間の制御の移行は，起動元のオブジェクトが，起動されるオブジェクトのオブジェクト名を，実行終了前に，図2 に示す次オブジェクト名記憶部に設定しておくことにより，メイン制御部によって行われる．説明を簡単にするために，フリー領域が充分にあるとすると，各オブジェクトの制御移行に従って，各オブジェクトは，メモリ上に，図4 (ロ) 図示のように展開されていくことになる．領域不足により，ガーベジ・コレクションが行われない限り，一旦，ロードされたオブジェクトは，メモリ上に存在するので，再び呼び出された場合には，再ロードすることなく使用することができる．もちろん，各プログラム・タイプ・オブジェクトは，予めリロケータブルに構成される必要がある．

3.2　教育用ソフトウェアの概要

以上の方式に従い，『アナグラム・ゲーム』[2]という教育用ソフトウェアを作成した．これは，一般にアナグラムと呼ばれる言葉遊び（バラバラに並んでいる文字や単語を，正しい順序に復元する処理処理を行うもの）をベースとして，楽しみながら知識の定着（語彙の習得等）を図ることを目的としたものである．実際の『アナグラム・ゲーム』の構成（各モジュールの呼び出し関係）を図5 (イ) に示す．

次に，図5 (ハ) によりアナグラムの操作例を示す．この例では，「ひがのぼる。」が問題素材となっている．この問題素材を（乱数等を使用し）例えば文字単位でバラバラに並び換えたものが問題となる．問題の文字または部分文字列をキー入力によって並び換えていくことにより，最終的に問題素材の並びと同じものが得られると，正解に到達したことになる．並び換えの移動の単位は，文字または部分文字列のかたまり（以下，チャンク）であり，一度結合された文字または部分文字列は，以後，分割されることはない．ここで問題の初期状態は「の」，「る」，「が」，「。」，「ぼ」，「ひ」であり，各々チャンクC1～C6となっている．ここで，チャンク「ぼ」を「の」と「る」の間に入れる指示があると，問題素材と問題の並びが照合され，「の」と「ぼ」と「る」が繋がって新しいチャンクC1となる．次に，チャンクC1～C4が新しい問題とされ，同様にチャンクの移動指示に対し結合処理が行われる．最終的にチャンクが一つになると完成となる．

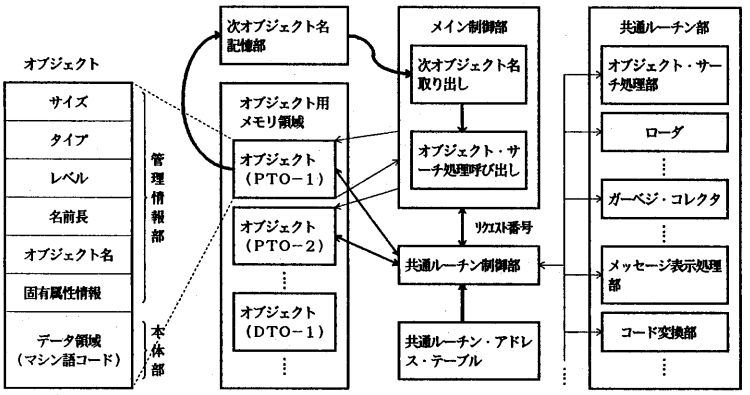
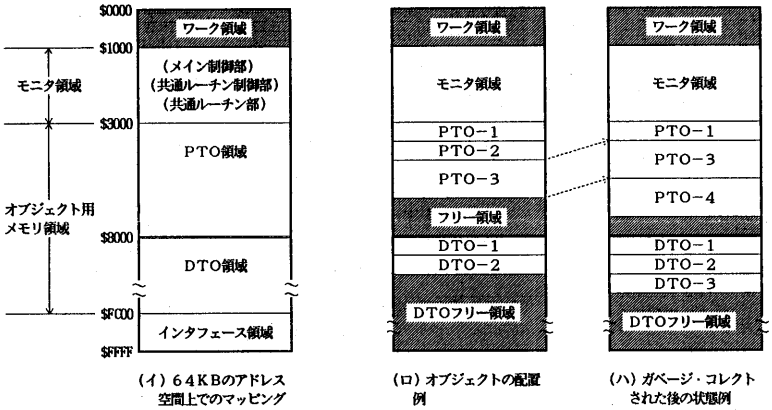
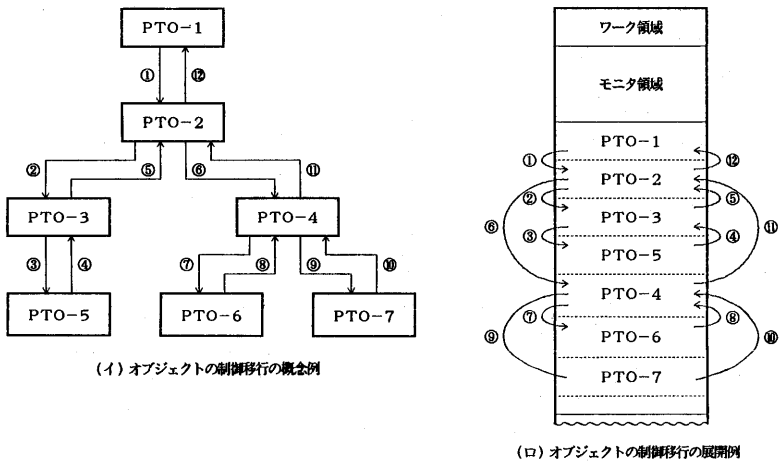


図2. オブジェクト実行制御機能の基本構成



(イ) 64KBのアドレス空間上でのマッピング　(ロ) オブジェクトの配置例　(ハ) ガベージ・コレクトされた後の状態例

図3. オブジェクト制御におけるメモリ・マッピングとその例



(イ) オブジェクトの制御移行の概念例　(ロ) オブジェクトの制御移行の展開例

図4. オブジェクト制御移行例

ここで，図5 (ロ) にその原理構成を示し，簡単に説明する．チャンク対応関係情報記憶部(Meta-A)には，現在の問題におけるチャンクと，問題素材データ記憶部中の正解との対応情報が保持され，チャンクの移動があった場合に，先頭結合処理部，後尾結合処理部，中間結合処理部によって，移動に関連した位置についてだけの結合処理がなされる．これにより，必ず一つの解だけが求められることになる．即ち，全ての解の検索を行わずに一つの有効な解が得られる方式を提供することにより，処理時間を短縮し，かつメモリ消費量を少なくすることを目的としている．

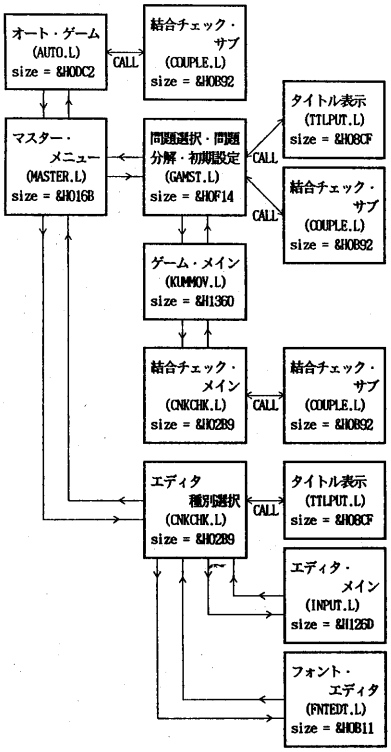
3.3 実施後の所感および評価

速度的にも満足でき，メモリ容量の壁も乗り越えることが出来たことの他に，次の①〜③の効果もうまれた：

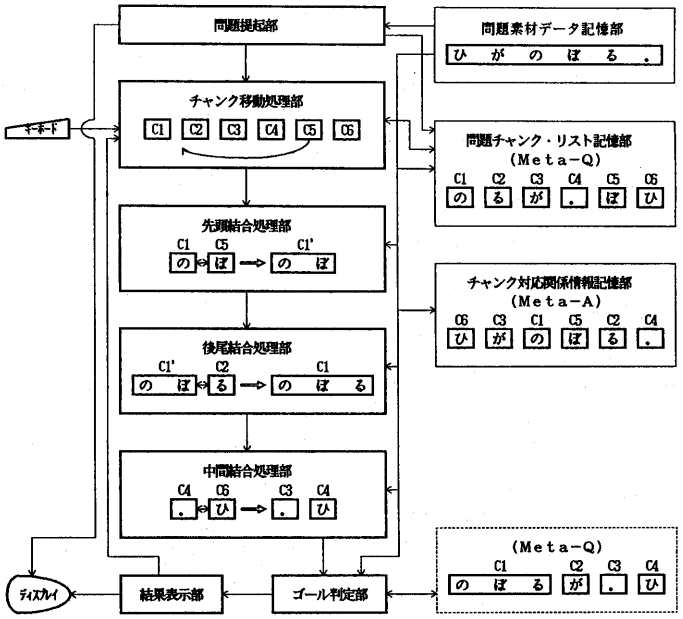
① 機能の新規追加（新しいオブジェクトの作成）が，システムの既存の部分に与える影響が少ないこと［すなわち，システムを開発している際，ある一部のプログラムのサイズが変化したために，そのアドレスより下にある他のモジュールのアドレスを変更するというようなことが不要になる．また，モジュールとモジュールの間に予備としてのメモリ（スペース）を確保しておく必要もなくなること］，

② 結合デバッグという考え方がほとんど不要になるので，デバッグし易いこと，

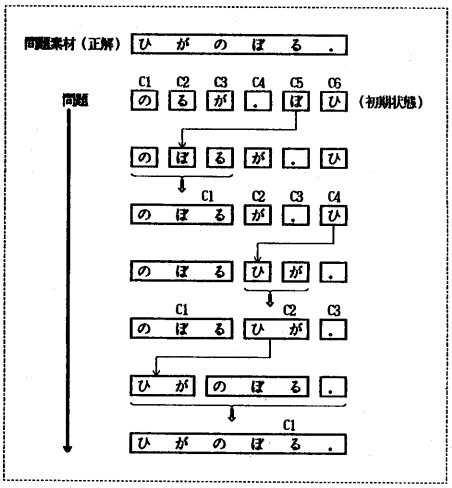
③ 複数の開発者がパラレルに作業を進めるので，開発が非常にやり易くなること．



(イ) オブジェクト制御方式におけるアナグラム・ゲームの構成



(ロ) アナグラム・ゲームの原理構成



(ハ) アナグラム・ゲームの課題説明

図5. オブジェクト制御方式における教育用ゲームソフトウェアの例

4. 階層メニュー構造プログラムの構築支援方式による Mind での開発事例

4.1 階層メニュー構造プログラムの構築支援方式の概要

この階層メニュー構造プログラムの構築支援方式は，所定の形式でデータを作成するだけで容易に（階層メニュー構造）プログラム構築を可能とすると共に，任意の文字列表示および単体のプログラムの確率的な実行制御を導入することにより，（柔軟なドリル型）教材ソフトウェアを作成するための手段等を提供することを目的としたものである．なお，この方式に関しては，教育用ソフトウェアの新規開発という目的よりは，むしろ既存のソフトウェアをどう整理し活用していくか，という視点(MMIの構築）があったことを予め述べておく．

図6 により，この方式の原理構成を簡単に説明する．メニューファイルには，メニュー表示用データおよび次に呼び出すファイル名データが所定のデータ形式で予め格納される．階層制御表示処理部は，現在表示しているメニュー中の任意の項目が（キーボードより）選択された場合に，その項目に対応するメニューファイルからメニュー表示用データを読み込み，各メニュー表示データについてディスプレイ表示を行うものである．実行制御処理部は，階層制御表示処理部において現在表示しているメニュー中の任意の項目が選択された場合に，その項目に対応する実行ファイルから実行データを読み込み，その指定された“繰り返し数”および“確率”等で実行サブファイルの実行を制御処理するものである．実行サブ制御処理部は，実行制御処理部における実行ファイル中に格納された実行サブファイルから実行サブデータを読み込み，文字列表示や単体プログラム（既存の教育用ソフトウェア等）の実行の制御処理を行うものである．

4.2　教育用ソフトウェアの概要

この方式では，ある特定の形態をもつ教育用ソフトウェアはない．例えば，ごく小規模のドリル型の教材を相当数用意しておき，それらをランダムに出題するといったようなものである．

図7 は，『中 3数学ドリル演習』といった教育用ソフトウェアを作成した例において，各データ（ファイル）どのように呼び出しあい階層的な関係を保っているかということを示している．

4.3　実施後の所感および評価

この方式の使用後の評価をまとめると，以下の①～③のようになる：

① 単体のドリル教材（ソフトウェア）を任意の確率で出題するような教育用ソフトウェアを簡単に作成できること（教育というような分野だけでなく，デモンストレーション用ソフトウェアといった発展方向も考えられる），

② 教材としての中身を入れ換えたい場合に，各データファイルの更新および別の単体のドリル教材の準備だけでよいから，効率がよいこと，

③ プログラム自身をソース提供するといった場合にも，こうした枠組み（ツール）を予め作成しておくことは，②との関連を考慮した上で有効であること（この方式に基づくツールは，Mind[5),6)] により開発しており，この実感は筆者らの経験[7),8)]に基づいている）．

なお，③に関連して，Mind によるプログラム例（一部）を図8 に示す．

5.　むすび

ここで報告した内容が良い悪いといったことではなく，CAI（CAL）に関しては，教育とは直接関係のないと思われる技術的な課題が必ずつきまとう．

安易な枠組み（ツール）の追求という形だけではなく，例えば海外の良い考え方[9)]にも耳を傾けるべきだし，今後さらに多方面，多次元での CAIに関する論議が活発化することを望む．[10),11),12)]

謝　辞

今回の報告内容に関し，㈱富士通大分ソフトウェアラボラトリ・システム部第三システム課の吉田泰明氏，進来玄一郎氏，徳広雅俊氏，大城文明氏，尾形卓治氏，十時伸氏，広瀬昭文氏，白石秀徳氏，滋野信二氏ら諸氏の努力に感謝致します．

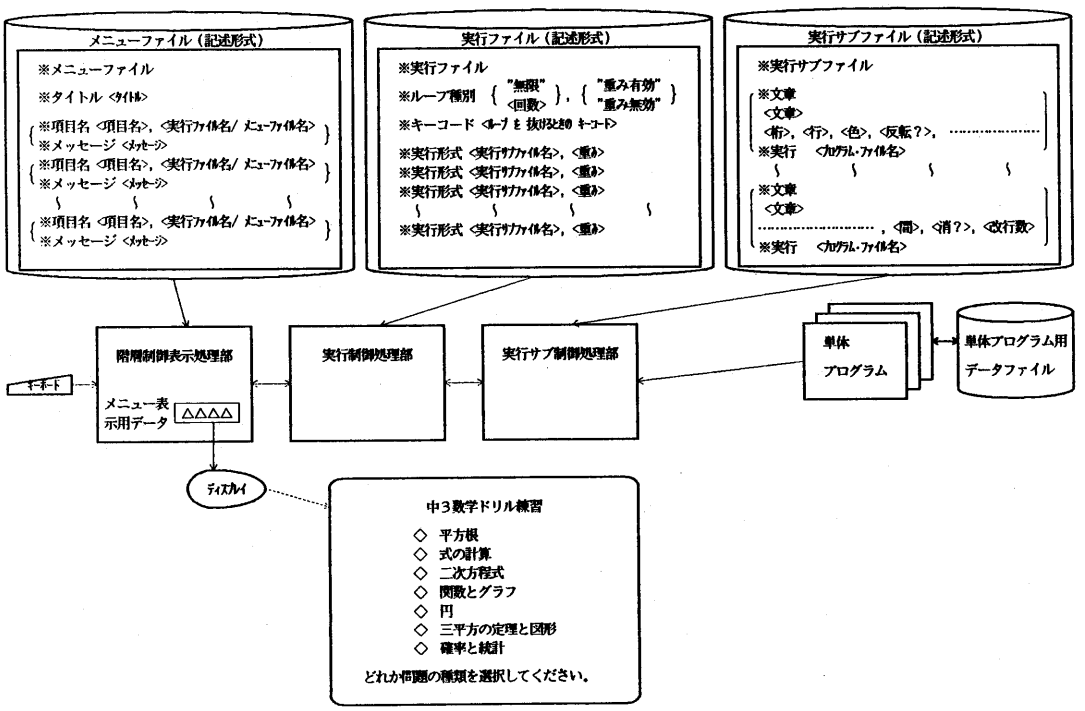


図6. 階層メニュー構造プログラムの構築支援方式の原理構成

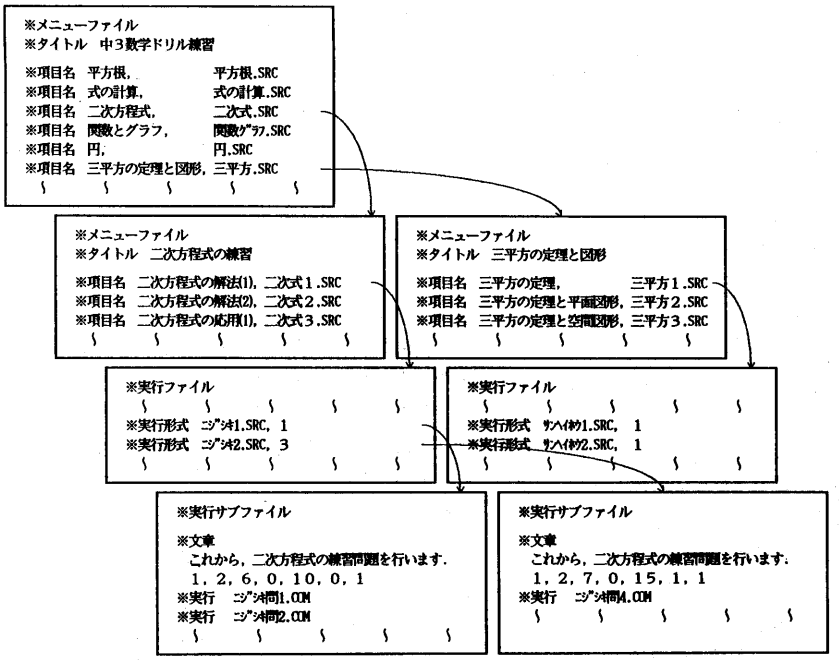


図7. 階層メニュー構造プログラムの構築支援方式における
メニューデータ，実行データ，実行サブデータの階層的な関係例

```
メニュー処理とは　（メニューファイル名　－－＞　．）
                        次回ファイル名１は　　文字列
                        単語１は　　文字列
                        桁位置１は　　変数
                        選択番号１は　　変数
    [階層　：＝　階層　＋　1]
    [階層　＞　最大階層]
        ならば　エラーメッセージ（６）を　ガイド表示し　二秒待ち
                [階層　：＝　階層　－　1]　終り
        つぎに
    メニューファイル名（階層）に　入れ
    ここから
        画面クリアし
        メニューファイル名（階層）で　シーケンシャルファイルオープンし
        エラーフラグオン？
            ならば　エラーメッセージ（７）を　ガイド表示し　二秒待ち
                    エラーフラグを　クリアし　打ち切り
            つぎに
        ファイルから　一行読み出し　捨て　（インデックス行を捨てる）
        [現行数　：＝　1]
        [項目数　：＝　1]
        24を　回数指定し
            ファイルから　一行読み出し　行に　入れ
            行から　単語を切り出し　単語１に　入れ
            単語１が　空列？　ならば　現行数を　一つ増加し　もう一度　つぎに
            単語１で　文字列事例をとり
                「※項目名」なら　項目処理し
                「※タイトル」なら　タイトル処理し
                「※メッセージ」なら　メッセージ処理し
                例外なら　記述エラー処理し
            事例終り
            エラーフラグオン？　ならば　エラーフラグを　クリアし　打ち切り
                                つぎに
        繰り返す
        [項目数　：＝　項目数　－　1]
        ファイルを　クローズし
        中断フラグを　クリアし
        メニュー表示し
        項目選択し　次回ファイル名１に　入れ
        次回ファイル名１が　空列？　ならば　打ち切り　つぎに
        次回ファイル名１の　ファイル種別を得て　単語１に　入れ
        単語１が「※メニューファイル」と　等しい文字列
            ならば「次回ファイル名１で　メニュー処理し
            つぎに
        単語１が「※実行ファイル」と　等しい文字列
            ならば　次回ファイル名１で　実行処理し
            つぎに
    繰り返す
    [◆の位置（階層）　：＝　1]
    [階層　：＝　階層　－　1]．
```

図8. 階層メニュー構造プログラムの構築支援方式における
Ｍｉｎｄによるプログラム例（一部）

参考文献

1) 平賀，芹沢，他 4名：教育用ｹﾞｰﾑｿﾌﾄｳｪｱ の開発経験の報告，富士通 '84SEｺﾝﾍﾞﾝｼｮﾝ 論文集（O03）(1984)．

2) 平賀，進来，他 3名：BX-easy の概念とその応用，微弱電波（富士通 CALｼｽﾃﾑ部内資料），No.12,17,20 (1985-1986)．

3) 進来，徳広，他 2名：ﾊﾟｿｺﾝﾒﾓﾘ の効率的な使い方，富士通 '86SEｺﾝﾍﾞﾝｼｮﾝ 論文集（B68）(1986)．

4) 富士通：FACOM OS IV / SPEAKEASY IV 使用手引書 (1983)．

5) 木村，片桐：日本語ﾌﾟﾛｸﾞﾗﾐﾝｸﾞ 言語 Mindについて，情処研報(88-PL-16)，Vol.88，No.27，pp.25-32 (1988)．

6) 片桐：日本語ﾌﾟﾛｸﾞﾗﾐﾝｸﾞ 言語Mind，翔泳社 (1988)．

7) 平賀：日本語ﾌﾟﾛｸﾞﾗﾐﾝｸﾞ 言語Mindによる CAL教材集，情報処理学会第36回全国大会論文集 (6K-2)，pp.2395-2396 (1988)．

8) 平賀：日本語ﾌﾟﾛｸﾞﾗﾐﾝｸﾞ 言語Mindによる ﾂｰﾙ集，教育工学関連学協会連合第 2回全国大会論文集 (3F04)，pp.583-584 (1988)．

9) Piestrup, A.: Game Sets and Builders— Graphics-based learning software, Byte, pp.215-219 (Jun. 1984)．

10) 佐伯：ｺﾝﾋﾟｭｰﾀと教育，岩波新書 (1986)．

11) 山本：ｽﾀﾝﾄﾞｱﾛﾝ 方式による CAI，電子情報通信学会誌，Vol.71，No.4，pp.379-384 (1988)．

12) 大槻，山本：知的CAI のﾊﾟﾗﾀﾞｲﾑ と実現環境，情報処理，Vol.29，No.11，pp.1255-1265 (1988)．