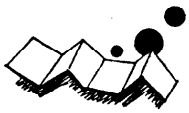
解　説



「SP-FLOW」によるデータ構造に基づくシステム設計法†

臼　井　義　美††

1. はじめに

近年，コンピュータのソフトウェアはますます高度化，大規模化しつつあるが，他方では OA 機器の普及などでさらに一般化し多様化に向かっている．ソフトウェアの開発の対象も，オペレーティング・システムや通信制御プログラムなどの専門的システムから，ビジネス・ソフトウェアなどの一般業務用システムまでさまざまな分野に広がっている．このうち一般業務用システムの開発の成否は，多くの場合，ソフトウェアの素人であるエンドユーザの要求を，いかに正確に効率良くシステム設計に反映するかにかかっており，システム設計に携わるすべての人々が容易に理解できる設計方法が求められている．また，このような業務用システムの開発に当たっては，設計の基礎として入出力情報を用いることが多いため，プログラム構造を決定するための基準としてデータ構造を利用するのが便利であると思われる[1),2)]．

本稿で紹介するシステムの設計法「SP-FLOW」は，設計者だけでなくエンドユーザをできるだけシステム設計に参画させ，目的のシステムを効率よく作成しようとするもので，要求分析から詳細設計に至るすべての工程に一貫した図式を利用することによって，データ構造に基づいたシステムを容易に構築することをねらっている．

以下，与えられた共通問題に沿って，SP-FLOW によるシステム設計の考え方と実現方法およびその特徴について解説する．

2. SP-FLOW の概要と記述法

2.1 概　要

本来，多種多様なシステム要求を効果的にシステム化するためには，その要求に合った最も合理的なシステムを設計すべきである．そのためには，対象となるシステム要求を自由に分析し，いろいろな代案の中から最も合理的なシステムを選び出し，具体化することができるような設計法が必要である．そして，最終的にはだれが設計しても同一の様式の設計書になり，設計者以外の人にも理解しやすいものが望ましい．

このため，SP-FLOW ではデータ構造とプログラム構造を記述する簡単な図式を用いることによって，自然語による記述のわかりやすさと論理の厳密さを両立させている．また，設計書の重要な記述要件であるデータ構造とプログラム構造，およびデータの流れと処理の流れを，常に比較対照しながら設計することができるような専用の設計様式を利用している(図-6 を参照)．この様式は上から下へ流れる時間軸を基準線として，その右側に処理の手続を記述するプロセス・ゾーンを，また，左側にデータに関する情報を記述するインタフェース・ゾーンを設けたものである．そして，双方の記述ゾーンに構造化のレベルを表わす縦線を設定することにより，構造化レベルを固定できるため，データ構造とプログラム構造の把握が容易となる．この方法によれば，設計者はデータ構造を示す図形の一部をプログラム構造を表わす制御フロー上に積木のように重ねていくことによって，容易に目的のプログラム構造を得ることができる．

2.2 データに関する記述

(1)　データ構造と内容の記述

本技法で使用するデータ構造図は，**図-1** に示すような構造化レベルを有するデータボックスと呼ぶ形式を利用する．

このデータ構造図は，繰返しや条件を示す構造化レベルを明示するとともに，従来のデータフォーマットの機能をも兼ね備えている．

(2)　データの流れの記述

データの流れを，システムの外部環境からプログラム内部まで，連続した動きとして記入できれば便利で

† System Design Procedure Based on Data Structures with "SP-FLOW" by Yoshimi USUI (Japan Information Processing Service Co., Ltd.).

†† 日本電子計算(株)

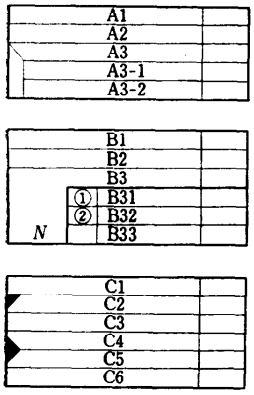


図-1 データ構造図

ある．本技法では，プログラム内部におけるデータの流れを，処理内容に対応するデータエリアに記入することによって，データの加工状況を制御構造と対比したり，処理の内容を視覚的に補足するのに役立てている．また，システムの基本的な仕様を検討するために，単にプログラムの入力データから出力データへの変換の過程のみをとらえるだけでなく，そのシステムを利用する環境を含めたデータの流れをもデータフローによって表現する．

2.3 処理に関する記述

制御の流れは，時間に沿った一本の線で表現することができるので，SP-FLOW では繰返しレベルと条件レベルで位置が確定する制御フローによって記述する（**図-2**）．

これによって得られる制御フローは，プログラムの制御構造を視覚的に表わし，繰返し処理や複雑な条件の組合せをも明確に表現できる[3]．

2.4 データ構造とプログラム構造

データ構造に基づく構造化プログラムを構築するためには，データの構造とそれに対応するプログラム構造の関係を明示できるような記述が必要である．

例えば，Jackson は**図-3** に示す図式によって両者の関係を表現している[2]．

同じ内容を SP-FLOW で記述したものが**図-4** である．データ構造図は内部を区分されたデータボックスとして図式化し，プログラム構造図は制御の流れる道筋を図式化しているため，具象的で理解しやすいものと考える．図-4から明らかなようにデータ構造図とプログラム構造図では，それを構成する繰返しや条件の構造化レベルを示す縦線の位置が完全に対応している．したがって，データ構造図からすべての横線を取り去れば自然にプログラム構造図の原型が得られる．

SP-FLOW におけるデータ構造図とプログラム構造図の対応関係をもう少し具体的に説明する．

図-5 に示す例ではAファイルを入力して B, C, D, Eファイルを出力するものとして，データ構造とプログラム構造の繰返しレベルを対比したものである．Aファイルの第1レベルの繰返しは，そのファイルがN1件のレコード A1 から成っていることを示し，こ

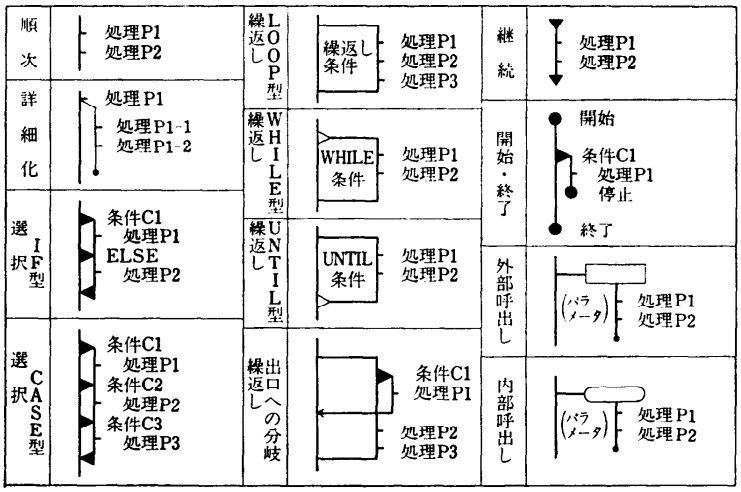
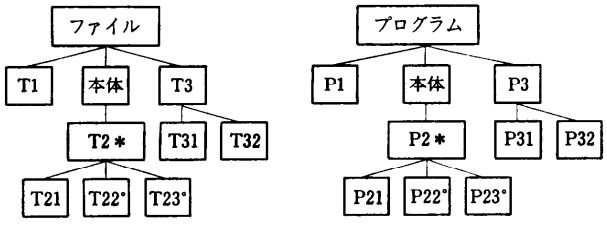
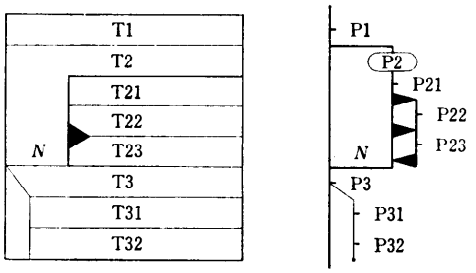


図-2 制御フロー



（a） データ構造図　　（b） プログラム構造図

図-3 Jackson の記述法（＊は繰返し，° は選択を示す）



（a） データ構造図　　（b） プログラム構造図

図-4 SP-FLOW による記述法（Nは繰返し回数を示す）

のレコード A1 はプログラムの第1繰返しレベルに対応する制御フロー上で入力される．また，A1 で示される各レコードには $N2$ 個の繰返し項目 A2 が含まれ，データの構造化レベルと同じ第2レベルの制御フロー上で処理される．そして，このレベルでBファイルを出力すると，Bファイルのデータ構造図は制御フローの実行回数（$N1 \times N2$）に等しいレコード数を含むことが示される．

このようにデータ構造のレベルとその繰返し回数（件数）は，プログラム構造のレベルと制御の繰返し回数（実行回数）に一致している．そして，データ構造図では，レコードの件数が構造化レベルを示す図形の幅で表わされ，プログラム構造図上では関係するレコードや項目を取り扱う制御線のレベル差に対応する．

一方，処理の内容は，対象となるデータ項目を含む構造化レベルに対応する制御フロー上に記述される．ただし，ある繰返しレベルに属するデータ項目の内容を，その上位または下位レベルのデータ項目に転送する場合がある．上位の繰返しレベルへの転送は，入力データの内容を集計したり，条件文をともなって入力データの特定の内容を抽出することを示し，逆に下位レベルへの転送は，入力データの内容を下位レベルの出力データ項目に配分したり，条件文と組み合わせて特定の項目に振り分けることを示す．

このように，この図式によれば，処理内容と構造化レベルの関係をより明解に表現することができる．

3. 要求定義

要求定義書は，開発しようとするシステムが何をするのかという利用者の要求を明確化していくものであ

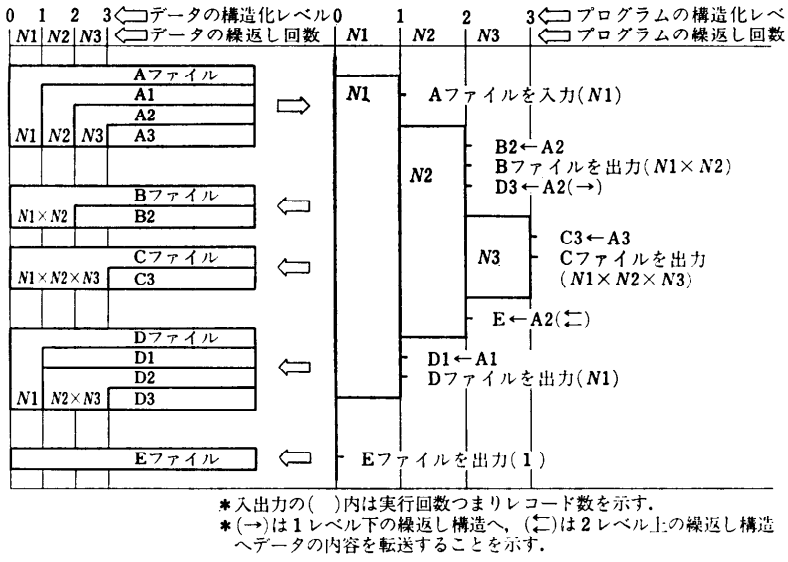


図-5 データ構造とプログラム構造の対応

るため，エンドユーザにわかりやすいものでなければならない．また，この段階で機能階層を取り入れると部分的な記述が多くなり，システム全体の内容が見えなくなる恐れがある[4]．SP-FLOW は，特定の設計様式と自然語を用いることによって，人間系の作業手順と，機械系に要求するシステムの処理内容やそれらの制限事項などを，利用者と設計者の双方で具体的に検討するのに便利である．

図-6 に共通問題に基づく要求定義書の例を示した．問題では積荷票についての処理を求めていないようであるが，ここでは，入庫と出庫の処理を1つのシステムとしてとらえ，設計に含めることにした．題意より，受付係の主な仕事は，商品の入庫にあたって積荷票を受け取ることと，出庫依頼に従って在庫商品の有無を確認し，出庫指示または在庫不足の連絡をすることである．まず，入庫に関する処理について，システムの外部環境における積荷票の流れと，それがシステムに入力されてから在庫ファイルに登録されるまでの過程をデータフローで記入する．同時に，積荷票の入力から在庫ファイルへの登録までの間に，システムに要求する処理の内容を，データフローに対応したプロセスゾーンに記入する．一方，出庫についても同様にシステム全体のデータフローと処理したい内容を記述する．なお，本例の説明では，プログラム内部のデータの流れを記述するデータエリアを入力エリア，在庫エリア，ワークエリア，作表エリアに割り当てた．

4. 基本設計

要求定義書ではソフトウェアが何をするのかを記述したのに対し，基本設計書はそれをいかに行うかを記述するものである．この方法では，自由な表現で記述された要求定義書の内容を，同じ様式のままシステム設計書に具体化していくことによって，エンドユーザを厳格な形式で記述されたシステム設計書の領域へと，自然に導くことができる．以下，前記の要求定義書をもとに，**図-7** に示す基本設計書を得る手順について説明する．

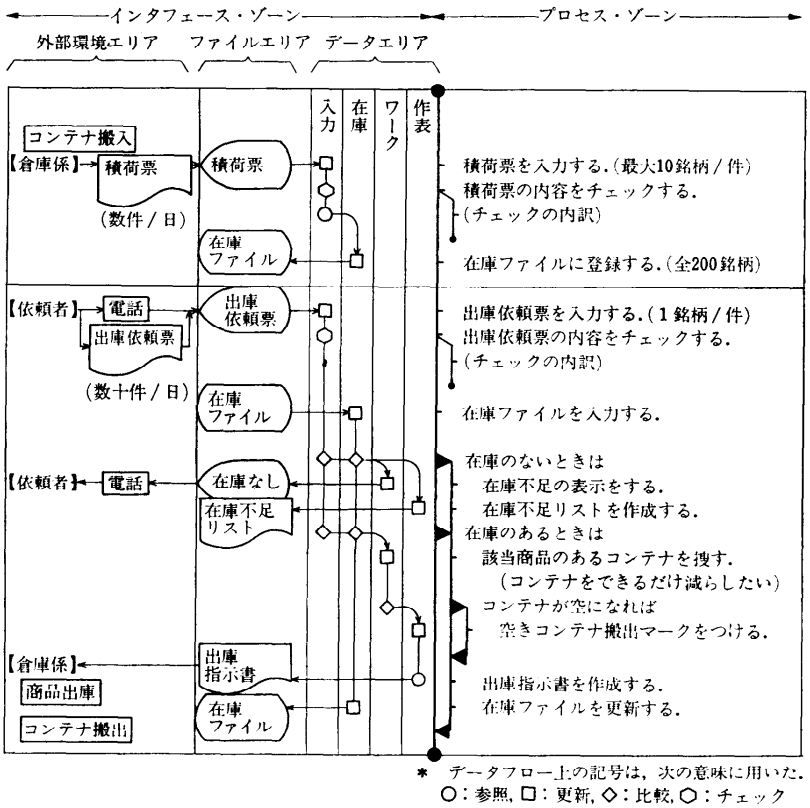


図-6 要求定義書の記入例

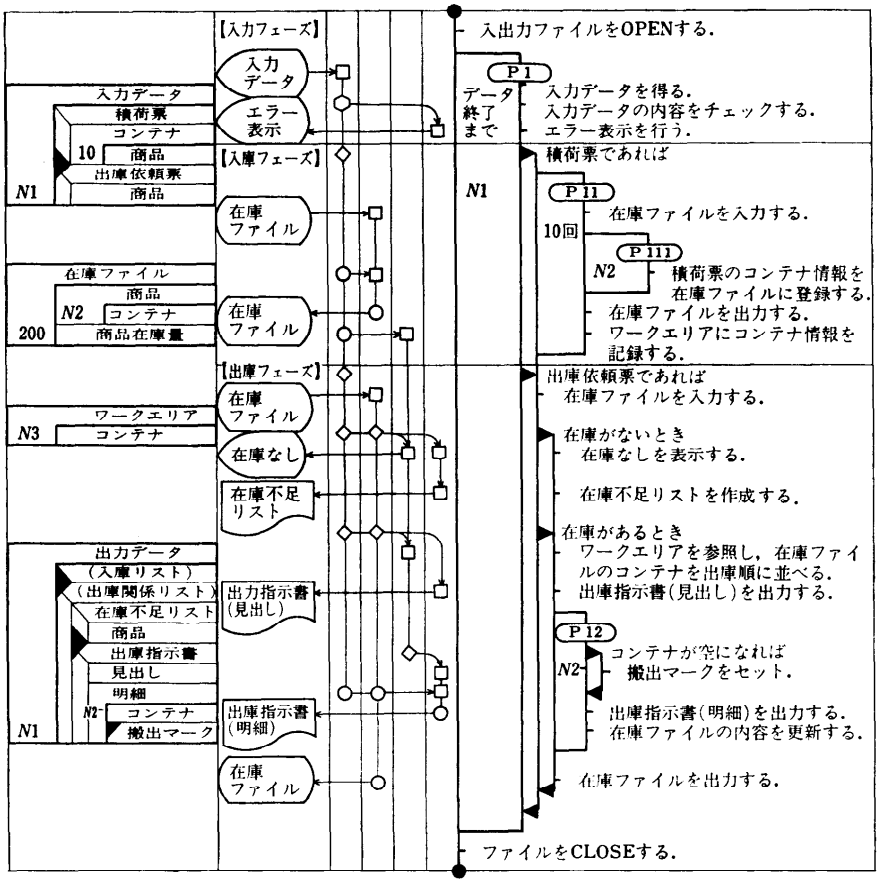


図-7 基本設計の記入例

4.1 データ構造図

入出力データの物理構造は，問題に与えられているので，そのまま図式化することにするが，基本設計ではデータの基本構造のみを取り上げる．

まず，入力データは，コンテナを搬入した時に受け取る積荷票と，酒類の出庫を依頼された場合の出庫依頼票から成り，同時に複数件の入力を認めるものとして，一度に処理するデータの合計を $N1$ とする．そうすると，ある入力データは積荷票か出庫依頼票のいずれかで，1枚の積荷票は最大 10 銘柄の内蔵商品の繰返しとなる．

一方，出力データには，出庫依頼票に対応する出庫指示書と在庫不足リストが設定されているが，積荷票に対応する入庫リストにあたるものを仮定すると，出力データの件数は入力データの件数 $N1$ に等しい．したがって，問題で要求された出力データの件数は，出庫指示書と在庫不足リストに関するものだけであるから，入力データ $N1$ の部分集合となり，$N1^-$ と表記する（**図-8** を参照）．

ところで，積荷票によって処理の対象となるコンテナが入力されると，出庫依頼票で出力の指示があるまで，コンテナに内蔵された商品の在庫状況をファイルとして保管する必要がある．この在庫の状態を管理するために在庫ファイルを定義する．在庫ファイルの構造は，入力データと同じコンテナ単位で記録するか，出力データに対応する商品単位に記録する方法の2通りが考えられるが，ここでは後者を採用した．

4.2 プログラム構造図

プログラム構造の導出については，文献1)，2)に詳しいが，SP-FLOW ではデータ構造図に現われる構造化レベルを抽出し，これを対応する制御フローの構造化レベルを決定する基礎として用いることによってプログラム構造図を得ることができる[5]．

(1) 共　有

入力データと出力データの構造が等しい場合は，同一のプログラム構造を共有することでそのデータに対する処理が可能である．例題では，入力データと出力データに $N1$ という基本的な繰返しの対応関係を見い出したのでこれをプログラム構造の共有部分とし，P1 というモジュール名の繰返し構造で表わす．また，入力データにおける積荷票と出庫依頼票の選択構造も，出力データの選択構造と共有できるので，データの構造化レベルに対応した P1 の構造化レベル上に配置する．

（2） 積 重 ね

入力データと出力データの構造が異なるときは，双方のデータ構造から導かれるプログラム構造を制御フロー上に積み重ねることで解決する場合がある．つまり，ある繰返し構造の入力データを異なる繰返し構造の出力データに転送する場合，入力データの繰返しに対応する制御フローの繰返し構造上に出力データの属する繰返し構造より得られた制御フローを書き加えることで，入力データに対するすべての転送先の出力データをサーチすることができる．もちろん，サーチのための繰返しは必ずしも最大実行回数とは限らず，要件を満たせば終了するし，転送先のデータが直接アクセス可能であり，1回の検索で完了する場合は，当該繰返し構造を省略することができる．

本例では，まず，積荷票の商品情報を在庫ファイルに登録する場合に適用する．すなわち，1コンテナ単位の積荷票には最大 10 銘柄の商品が記録されているので，この商品を取り出すために P1 のレベルの積荷票側に P11 という最大10回の繰返しを設定する．そして各商品を在庫ファイルに登録するために，在庫ファイルのデータ構造図より得られる 200 回の繰返し（在庫ファイルのレコード数＝商品の銘柄数）と $N2$ の繰返し（特定の商品が存在するコンテナ数）をP11の上に積み重ねることによってデータ構造の変換を行い，在庫ファイル中にコンテナ情報を登録する．もし，在庫ファイルを商品コードで直接アクセスできるように設計すれば，図-7 に示すように 200 回の繰返し構造が不要となる．

一方，出庫依頼票に対する出力データは，在庫不足リストと出庫指示書の二者択一となるため，その選択構造を出庫依頼票に当たるプログラム構造上に積み重ねる．さらに出庫指示書の商品は，在庫ファイルに保存された $N2$ 件のコンテナをサーチして得ることとし，P12 で示される繰返し構造を出庫指示側に積み重ねる．最後に，出庫指示書に空コンテナマークが必要な場合があるため，コンテナをサーチする P12 の制御フロー上に選択構造を記述する．

（3） 分 割

入力データと出力データのデータ構造が異なり，「積重ね」による解決が困難なときは，中間ファイル等を利用してデータ構造を変換する．この場合は，入力データから中間ファイルを出力するプログラム構造と，中間ファイルから出力データを得るプログラム構造を，それぞれ「共有」または「積重ね」の方法を用いて決定する．

前述のように在庫ファイルは商品単位に記録する方法を採用したので，空コンテナを搬出するために，在庫ファイルからコンテナ単位のファイルを中間的に作り出す必要がある．ここでは，倉庫に保管できるコンテナ数 $N3$ のワークエリアを設定した．したがって，コンテナ情報をワークエリアに登録するために，積荷票の内容を展開する P11 の制御フロー上に $N3$ 回の繰返しが必要であるが，ワークエリアの空きレコードをインデックスで検索できるものとすれば，この繰返し構造を省略できる．また，ソート済みのワークエリアからコンテナ情報を取り出す場合も，ワークエリアの内容はインデックスで順次抽出可能であるから，$N3$ の繰返し構造を省略することができる．

4.3 処理の手続きとデータの流れ

プログラム構造図を構成する制御フロー上に，実行すべき順序で処理の手続きを割り当てる．基本設計の段階では，通常の日本語として十分読めるような文章で記述する方が，要求者や利用者と処理内容を確認するのに便利である．また，処理の手続きを記入するたびに，データの移送や加工の状況をデータフローとして記入し，処理内容を再確認する．

4.4 モジュール化

モジュールを中身の見えない「ブラックボックス」の形で取り扱うことが多い[6]が，SP-FLOW では次に示すようにできるだけ制御フローや処理内容が見える「ホワイトボックス」としてモジュール化する．

（1） 制御構造によるモジュール化

このモジュールは，制御フローを決定することによって得られるもので，制御構造に基づいて下位モジュールの実行要件を決定する．SP-FLOW では，繰返しレベルと条件レベルの2つの構造化レベルで定まる線分として記述され，「横方向のモジュラリティ」を明示する．この線分上の命令群は，Warnier の指摘す

るように同一条件下で，同一回数実行される命令の集合として表わされる[1]．

（2） 処理工程によるモジュール化

プログラムは時間的な処理工程の推移によって，あるまとまった機能を順次行うのが一般的である．本技法では，処理フェーズという形で時間的な機能境界に沿って処理の内容を分割し，「縦方向のモジュラリティ」として表現している．このモジュールは制御レベルに関係なく単に時間の次元による分割であり，例えば「入力フェーズ」，「入庫フェーズ」のように処理の工程を横線で分断することによって示す．ただし，「入庫フェーズ」と「出庫フェーズ」のように時間的に並列である場合も考えられる．

（3） 機能の一般化によるモジュール化

これは，単一の機能を有する処理の一部分を，汎用ルーチンという形で「機能モジュラリティ」としてまとめたものである．このモジュールはいちいち内容を確認しないで一定の機能を果たす部品として取り扱うことができるものをいい，プログラムの制御構造に重大な影響を与える部分や，目的とする処理内容が不明確となる部分は含めるべきではない．

（4） プログラムとして取扱うためのモジュール化

システムの設計やテストに便利で運用に効率の良い大きさのプログラムとして取り扱うために，主に2種類の方法でモジュール化を行う．その1つは，プログラムを処理工程によって取り扱いやすい大きさに分割したり，並列処理や自由な順序で処理するためにプログラムを分ける場合で，一般に制御線が基準レベルに存在する部分で分割される．他の1つは，ある制御レベル上の特定の処理を別のプログラムとして作成し，外部モジュールの呼出しという形でプログラムを階層化する場合である．

4.5 最 適 化

本技法で行う最適化には，データ構造の最適化，制御構造の最適化，処理手順の最適化がある．

例えば，SP-FLOW で記述した制御フローはデータ構造に基づいて構造化されているため，データ構造を変更することによってプログラムが簡単な構造になる場合は，利用者の許す範囲で最も合理的な形式に変更すべきである．また，同じ処理内容があちこちに分散しているのをまとめたり，処理の順番を変えるなど，処理の実行を制御フロー上の最も適切な位置に再配置することによってプログラムの最適化を計る．

本例では，先に述べたように在庫ファイルが商品コードをキーとして直接アクセスできるように設計したので，このファイルを入出力するたびに必要な繰返し構造が不要となる．また，在庫ファイルに商品ごとの在庫量を登録しておけば，在庫の有無を判断するための繰返し構造を省略することができる．同様に，コンテナが空になったかどうかを判定するためのワークエリアについてもインデックスで直接アクセス可能とすれば，このエリアをサーチするための $N3$ の繰返しも不要となる．

4.6 代替案の選択

ここで説明した例では積荷票と出庫依頼票を複数件同時に入力する形で設計したが，積荷票の処理と出庫依頼票の処理を，それぞれ別の繰返しとして設計した場合は，それらの2つの繰返しの間でプログラムが分割される．また，入力データを1件ごとにリアルタイムに処理する場合は，入力データ，出力データの $N1$ の繰返し構造が不要となるため，対応するプログラム構造から $N1$ の繰返しを抹消することによって実現することができる．そのほか，在庫ファイルをコンテナ・商品順の構造とした場合は，ここに述べた手順でそのデータ構造に応じたプログラム構造を導き出す．このように設計者が作成したいろいろな代替案の中から，最も要求に近い設計案を選択する．

5. 詳 細 設 計

最適な基本設計書が確定すれば，同じ様式を用いて詳細設計書を作成する．この場合，基本設計書を時間軸に沿って縦方向に拡大することにより制御フローをさらに詳細化し，処理の内容を具体化する．すなわち，基本設計書に記入した処理内容に詳細化記号を付加し，その上に具体的な処理の内容を記述していく．この方法は，縦方向のズーミング機能によるもので，基本設計書と詳細設計書のつながりが自然であるため，詳細化が容易である．したがって，詳細化の過程で論理の書換えをともなわないため，誤りの発生を防ぐことができる．また，処理内容が実行される順に記述されているため，コード化がそのままの形で行え，プログラムと設計書が完全に対応したものとなる．

詳細設計書の例を図-8 に示す．図のように，詳細化は多くの頁に継続して記述されるが，構造化レベルの位置を固定しているため改頁が設計上の大きな障害とならない．むしろ，処理フェーズごとに頁を分けることによって，頁単位の挿入・削除が容易となり，設計内容の変更が行いやすくなることが多い．

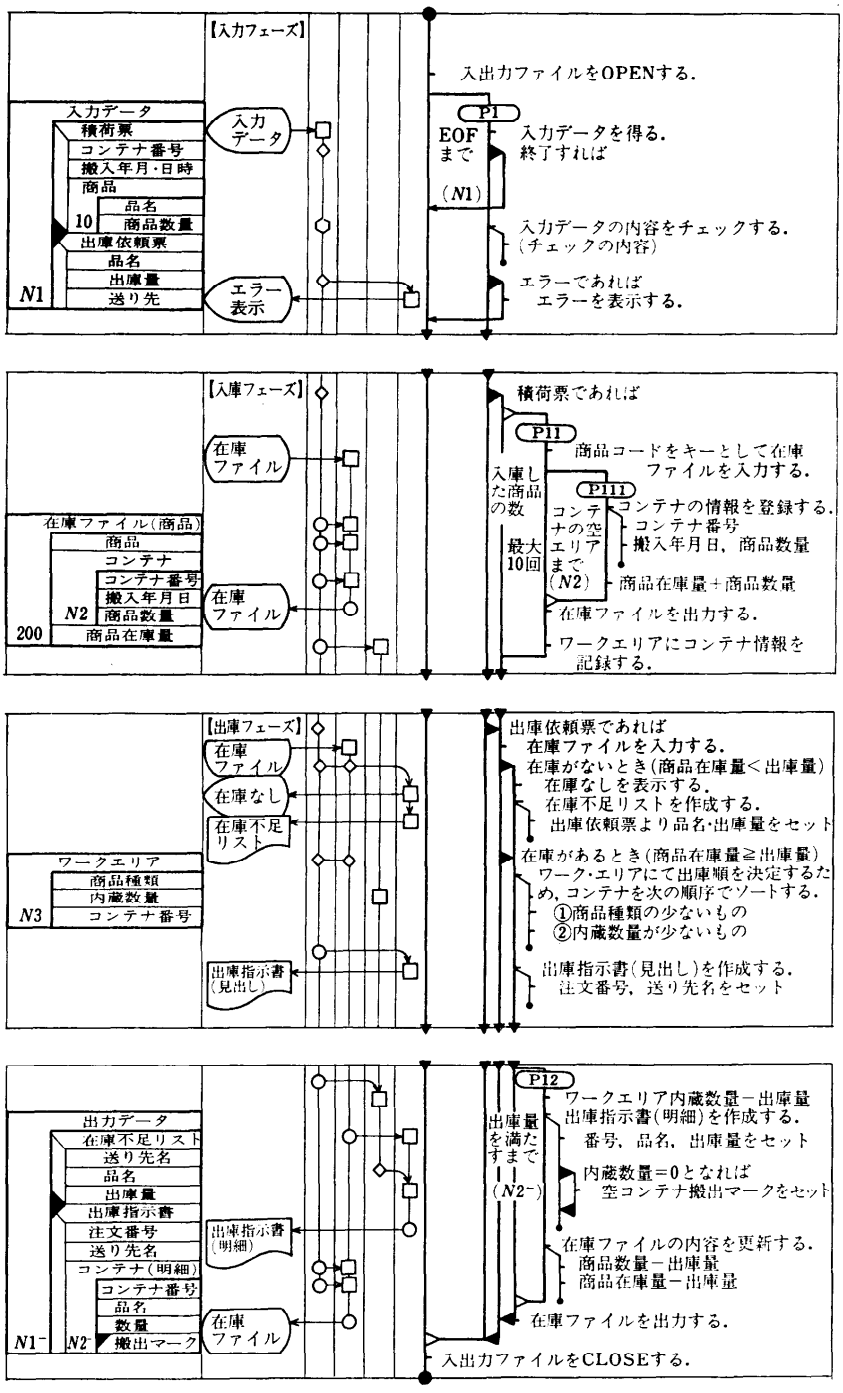

【入力フェーズ】
入出力ファイルをOPENする.
入力データ
積荷票
コンテナ番号
搬入年月・日時
商品
品名
10
商品数量
出庫依頼票
品名
出庫量
送り先
N1
入力データ
エラー表示
P1
EOFまで
(N1)
入力データを得る.
終了すれば
入力データの内容をチェックする.
(チェックの内容)
エラーであれば
エラーを表示する.
【入庫フェーズ】
積荷票であれば
在庫ファイル
P11
商品コードをキーとして在庫ファイルを入力する.
入庫した商品の数
P111
コンテナの空エリアまで
(N2)
最大10回
コンテナの情報を登録する.
コンテナ番号
搬入年月日, 商品数量
在庫ファイル(商品)
商品
コンテナ
コンテナ番号
搬入年月日
N2
商品数量
200
商品在庫量
在庫ファイル
商品在庫量+商品数量
在庫ファイルを出力する.
ワークエリアにコンテナ情報を記録する.
【出庫フェーズ】
出庫依頼票であれば
在庫ファイルを入力する.
在庫がないとき(商品在庫量<出庫量)
在庫なしを表示する.
在庫不足リストを作成する.
出庫依頼票より品名・出庫量をセット
在庫ファイル
在庫なし
在庫不足リスト
ワークエリア
商品種類
内蔵数量
N3
コンテナ番号
在庫があるとき(商品在庫量≧出庫量)
ワーク・エリアにて出庫順を決定するため, コンテナを次の順序でソートする.
①商品種類の少ないもの
②内蔵数量が少ないもの
出庫指示書(見出し)
出庫指示書(見出し)を作成する.
注文番号, 送り先名をセット
出力データ
在庫不足リスト
送り先名
品名
出庫量
出庫指示書
注文番号
送り先名
コンテナ(明細)
コンテナ番号
品名
数量
N1⁻
N2⁻
搬出マーク
出庫指示書(明細)
在庫ファイル
P12
出庫量を満たすまで
(N2⁻)
ワークエリア内蔵数量−出庫量
出庫指示書(明細)を作成する.
番号, 品名, 出庫量をセット
内蔵数量=0となれば
空コンテナ搬出マークをセット
在庫ファイルの内容を更新する.
商品数量−出庫量
商品在庫量−出庫量
在庫ファイルを出力する.
入出力ファイルをCLOSEする.

図-8 詳細設計の記入例

6. おわりに

以上，SP-FLOW によるシステムの設計法について，その概要と実現方法を述べた．

最近では，オブジェクト指向型や関数型，論理型などの非手続き型言語を用いた新しい設計法が提唱されている中で，ここで示した方法は，どちらかと言えば古典的な設計法に属するものである．しかし，本技法は，データ構造図からプログラム構造を図式によって導出することができ，要求定義から詳細設計に至る開発過程において，一貫した記述様式で利用できる実用性の高い方法であると考える．また，データ構造とプログラム構造の最適化を追及することで，テスト容易性とシステムの信頼性を高めることができ，読みやすくプログラムとの対応性の良い設計書は，保守用ドキュメントとしても使いやすいものである．

共通問題の例で示すように，SP-FLOW は全体が見通せる程度の小規模なシステムの開発に特に効果的である．したがって，大規模なシステムに適用する場合は，処理フェーズによるシステムの分割や，外部モジュールを用いたプログラムの階層化を行うことによって，この方法を有効に利用することができる．筆者等の経験では，詳細設計にして 10〜20 頁程度，500〜2000 ステップくらいのプログラムの記述に最も適していた．

また，本技法で用いた図式表現は，すでに実用化している自動ドキュメンテーション[7]や，今後主流になるであろう画像処理を利用したソフトウェアの開発にも適したものであると思う．

参 考 文 献

1) Warnier, J. D. (鈴木君子訳)：ワーニエプログラミング法則集，日本能率協会 (1975)．
2) Jackson, M. A. (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)．
3) 臼井義美：「SP-FLOW」による構造化プログラム設計書の記述法，情報処理学会第 23 回全国大会論文集，pp. 405-406 (1981)．
4) 宮本 勲：ソフトウェア・エンジニアリング：現状と展望，TBS 出版会 (1982)．
5) 臼井義美：図式によるプログラム構造の決定方法，情報処理学会第 29 回全国大会論文集，pp. 549-550 (1984)．
6) Myers, G. J. (国友義久，久保未沙訳)：高信頼性ソフトウェア―複合設計，近代科学社 (1976)．
7) 臼井義美，坂上和子：構造化プログラムのドキュメント自動作成システム―AUTO/SPFLOW，情報処理学会第 24 回全国大会論文集，pp. 353-354 (1982)．

(昭和 59 年 9 月 3 日受付)