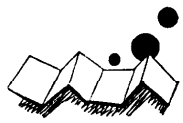
解　説



MOTHER SYSTEM によるソフトウェアの設計と製造†

西　村　高　志††

1. まえがき

MOTHER SYSTEM は情報処理振興事業協会(IPA)技術センターで，昭和56年10月より3年計画で進めてきた技術開発の最終成果物である．この技術開発は，当初には各企業に蓄積されているソフトウェア資産を再利用できるように活性化する技術の開発を目標としたが，任意に作られたソフトウェアを対象にすることは困難だと判断して，昭和57年7月に目標をソフトウェア部品化技術の開発に絞った．対象分野は事務処理 COBOL プログラムである．そして，給与計算プログラム[1]をモデルとして部品化技術の開発を進めた．昭和58年3月に部品化技術の概念設計をまとめたが，2カ月後開発目標をプログラム合成技術に再度変更して，プログラム合成技術を中核とする仕様部品からのプログラム製造技術として概念設計をまとめ直した[2]．システム開発は58年9月に開始して，59年3月に端末の日本語メニュ画面を介して外部仕様部品からプログラムを製造する実験システムを開発した[3),4)]．開発および稼動計算機は VAX-11/780，OS は UNIX/Berkeley 版 4.1，端末は日本語 JIS 第1水準表示機能を持つ KJ-100 である．プログラム合成部とデータベース操作部，メニュ画面制御部とはコンパイラ・コンパイラ YACC(Yet Another Compiler-Compiler) により作成した．システム制御部など YACC で作れない残りの部分は言語 C で作成した．

MOTHER SYSTEM はプログラム合成技術が実現可能であることを示し，さらに合成技術を中核としたソフトウェア製造形態がどうなるかをたしかめるために，対象を事務処理プログラムの限定された処理形態に絞って開発したシステムである．

† The Software Design and Production on the MOTHER SYSTEM by Takashi NISHIMURA (Software Technology Center, Information-technology Promotion Agency, Japan).

†† 情報処理振興事業協会技術センター

2. 合成されるプログラムの処理形態

MOTHER SYSTEM は，プログラム仕様に位置付けている3宣言——プログラム宣言とデータ宣言，手続き宣言——から単体で実行可能な COBOL メイン・プログラムのソース・テキストを合成する．

合成されるプログラム処理形態：

主キー項目値の昇順にレコードが並んでいる主ファイル（順編成）の各レコードについて——主ファイルが複数ある場合には，各主ファイルから集めた主キー項目値が同一値のレコード群について——，そのレコード値で決まる関連ファイル（順編成または索引編成）のレコードを入力する．そして入力したこれらのレコード値から出力ファイル(順編成)のレコード値を演算して，出力ファイルごとに指定された出力条件が成り立つならレコードをファイルに出力する．この操作を主キー項目値の昇順に主ファイルのすべてのレコードについて繰り返す（図-1）．

補足説明：

- ファイル数には制限はない．
- ファイル以外に演算の際に利用できる内部デー

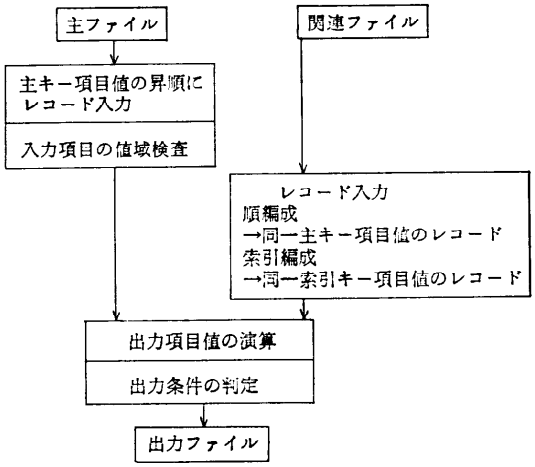


図-1　合成されるプログラムの処理形態

プログラムの機能：
給与トランザクションと給与マスタから給与明細を計算する．入力した給与トランザクションに誤りがあればエラー・リストにその内容を出力する．

プログラム宣言：
主キー項目： 社員番号
主ファイル： 給与トランザクション
関連ファイル：給与マスタ
出力ファイル：給与明細とエラー・リスト
給与明細の出力条件： 入力エラーなし
エラー・リストの出力条件：入力エラーあり

プログラムの処理形態：
給与トランザクションの各レコードを社員番号の昇順に読みながら，その社員番号の給与マスタのレコードを入力する．入力した給与トランザクションおよび給与マスタのレコード値より給与明細およびエラー・リストのレコード値を演算する．そして，入力エラーの有無により実際にレコードをファイルに出力する．

図-2 「給与計算」のプログラム宣言例

タを定義できる．

- 主ファイルと関連ファイルについては入力レコードが存在しない場合がある．その場合には入力レコード値としてレコード内各項目を COBOL の INITIALIZE 命令で初期化した値，つまり空白や零が取られる．さらに，レコードの有無を検査するために2つの**条件関数 $exist（ファイル名）** と **$empty（ファイル名）** とを提供している．前者は入力レコードが存在した場合に，後者は存在しない場合にそれぞれ真になる．
- 主ファイルの入力レコードの各項目に値域検査を指定できる．
- 関連ファイルの入力レコードを決める方法は関連ファイルの編成属性により異なる．順編成の場合には――関連ファイルも主キー項目値の昇順に並んでいる――主キー項目値が一致するレコードを入力する．索引編成の場合には関連ファイルごとに索引キー項目とこの索引キー項目と同一項目を持つ主ファイルとが指定されており，指定された主ファイルの入力レコードと索引キー項目値が一致するレコードを入力する．
- 出力ファイルのレコードの各項目は，演算に先だち COBOL の INITIALIZE 命令により初期化され，空白や零が代入される．

給与明細を計算するプログラムのプログラム宣言の一例とそのプログラム宣言から作られるプログラムの処理形態とを**図-2** に示す．

事務処理プログラムではプログラム処理パターンが分類されている．体系的標準集 STEPS/C のプログラミング標準では，このパターンを媒体変換と分類，抽出，集計，分配，併合，照合，更新，報告書作成とに分類している[5]．上記の処理形態はこの内で抽出と集計，分配，併合，照合，報告書データ作成とを満たせるように設計した．残りの媒体変換と分類，更新と報告書本体作成とを扱う能力を現在のシステムはもっていない．

MOTHER SYSTEM のプログラム仕様表現ではファイルよりもレコードを中心に扱う．そのために，以降の説明中ではファイルという名称を避けて，主データと関連データ，出力データとの名称を使う．

3. 合成されるプログラムの内部構成

MOTHER SYSTEM は日本電気 ACOS-4/MVP の COBOL 言語仕様のソース・プログラムを合成する（**図-3**）．

図-3 で手続き部 (PROCEDURE DIVISION) は，ファイルのオープンとクローズ，1レコード入力，出力レコード初期化とをそれぞれ分担する内部関数群と次の4部とからなる．

- 一連の操作を繰り返す**制御部**．
- 次の主キー項目値について主データと関連データとを入力する**入力部**．
- 一群の出力データ項目値を演算する**演算部**．
- 出力条件を検査して，条件が真なら出力データ項目をレコードに編成してファイルに出力する**出力部**．

COBOL コンパイラが COBOL 言語を機械語命令列に変換するのに対して，MOTHER SYSTEM はプログラム仕様言語を COBOL 命令列に変換する．変換は，上記のプログラム処理形態を**アルゴリズムの**

```
IDENTIFICATION DIVISION.
    プログラム名
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
    ソース／オブジェクト・コンピュータ名
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    ファイル編成と内部／外部ファイル名
DATA DIVISION.
FILE SECTION.
    ファイルのレコード構成宣言
WORKING-STORAGE SECTION.
    データ宣言
PROCEDURE DIVISION.
    制御部
    入力部
    演算部
    出力部
```

図-3 合成されるプログラムの内部構成

ひな形と見て，このひな形にプログラム仕様の内容をはめこむことで行われる．

プログラム仕様は次の3宣言よりなる．

- プログラム名と主キー項目名，データ構成，ファイル属性とを宣言する**プログラム宣言**．
- 各データの項目構成を宣言する**データ宣言**．
- 出力データおよび内部データの各項目値の演算を宣言する**手続き宣言**．

このアルゴリズムのひな形の特徴は，一群の入力レコード値がすべてそろった時点で一群の出力レコード値を演算している点にある．これにより，図-3 中で演算部以外のすべての部分をプログラム宣言とデータ宣言とから合成できる．手続き宣言は残された演算部を合成するために設けた．

4. MOTHER SYSTEM によるプログラム設計

MOTHER SYSTEM でのプログラム設計とは，開発対象プログラムの機能をこのアルゴリズムのひな形に合わせてプログラム仕様の3宣言を作ることである．そして，プログラム製造は「合成」に吸収されてしまう．

以下に課題プログラムをやや非現実なシステムにはなっているが，MOTHER SYSTEM により作成する．かたがたこのシステムではプログラムをどのように記述するかを説明する．なおシステムの入力文字の種類は英小文字を標準とした ASC-II コードであるが，説明の便宜上，名称を日本語に代えている．

4.1 プログラム宣言

プログラム宣言を決める上での重点事項は主データと主キー項目との選択にある．

(1) 主データと主キー項目の決定

課題プログラムは出庫依頼を入力して出庫指示書と在庫不足リストとを出力する．倉庫内のコンテナについては，説明の便宜上，順編成ファイルとして扱うことにする——コンテナ在庫と更新コンテナ在庫．

プログラム処理形態：

まず出庫依頼を入力する．そしてコンテナ在庫の各レコードを次々に入力しながら出庫依頼に従って更新コンテナ在庫のレコードを出力する．必要に応じて出庫指示書と在庫不足リストとを出力する．

- 主データ： コンテナ在庫，出庫依頼
- 主キー項目：コンテナ番号

出庫依頼は出庫依頼1レコードからなる順編成ファイルとし，さらに項目コンテナ番号を加える．出庫依頼を最初に入力させるために，項目コンテナ番号の値を出庫依頼は常に零に，コンテナ在庫は1以上にする．

```
主キー項目：コンテナ番号
主データ  ：コンテナ在庫
            出庫依頼
出力データ：更新コンテナ在庫  出力条件：
                              $exist（コンテナ在庫）
            出庫指示書        出力条件：出庫あり
            在庫不足リスト    出力条件：
                              在庫なしか数量不足
```

図-4 課題プログラムのプログラム宣言の概要

ここまでで決まるプログラム宣言を図-4 に示す．

(2) 内部データ

出力データを入力データから直接に演算するように説明していたが，現実にプログラムを記述するにはこのままでは無理がある．

- 一般には演算式が冗長で複雑になる．
- 次の入力データによる演算の時点まで演算結果値を記憶する機能がない．

これらを解決するために入出力ファイルに対応しない内部データを提供している．内部データには3種類の属性がある．

- **属性 temporary**：演算中間値の記録用であり，データ入力時点で初期化される．
- **属性 permanent**：状態や累計の記憶用である．これまでの入力データによる演算結果値**旧値**と今度の入力データによる演算結果値**新値**とが共存する．そして，新値は次の入力データに対する新たな旧値として引き継がれる．プログラム開始時の旧値としては，データ宣言で指定された初期値が取られる．
- **属性 lookuptable**：事務処理プログラムでよく使われる表引きデータ用である．参照のみ可能であり，値はデータ宣言で指定される．

図-4 のプログラム宣言に，出庫依頼の記憶用に permanent 属性の内部データ注文記録を追加する．注文記録の内容は出庫依頼と同じとする．

4.2 データ宣言

MOTHER SYSTEM で扱えるファイルは同一構造のレコード列からなるファイルに限定される．そのために出庫指示書は注文番号と送り先名とを含めた繰り返しとする．積荷票の（内蔵品名，数量）の繰り返し数は1コンテナに混載できる銘柄数の容量である10とする．出庫指示書中の注文番号は，倉庫受付係が出庫依頼を受け付けた時点で決めるものとして，出庫依頼に項目注文番号を追加する．

表-1 MOTHER SYSTEM での項目属性表現

属性種類	MOTHER SYSTEM	ACOS-4 COBOL での表現
文字列	char(*m*)	pic x(*m*)
整数	dec(*m*)	pic 9(*m*)
実数	dec(*m*, *n*)	pic 9(*m*) v 9(*n*)
符号付き整数	sdec(*m*)	pic s 9(*m*)
符号付き実数	sdec(*m*, *n*)	pic s 9(*m*) v 9(*n*)
単精度固定2進数	bin 1( )	usage comp-1
倍精度固定2進数	bin 2( )	usage comp-2
内部10進数整数	pdec(*m*)	pic s 9(*m*) usage comp-3
内部10進数実数	pdec(*m*, *n*)	pic s 9(*m*) v 9(*n*) usage comp-3

データ宣言では，各データごとに COBOL の構造体宣言の形式によりデータ内各項目の**レベル番号と項目名称，属性，初期値，値域，値域外処置**とを宣言する．**付録(2)**にデータ宣言を示す．データ**一時記録**については **4.3(6)** で説明する．

補足説明：

- 属性については，COBOL 属性表現中で usage 句の書き方がマシンに依存しているので，表現を全面的に変えている（**表-1**）．
- 入力項目の値域検査は，各項目ごとに指定された値域と値域外処置に基づき行われる．値域の書式は値域検査用外部手続き呼び出し（**$ 値域関数名（値域範囲パラメータ列）**）である．付録(2)中の**範囲**は数字値域を検査する外部手続き名である．値域検査用外部手続きと演算関数用外部手続きとは MOTHER SYSTEM の運用責任者により引数パラメータの順序と属性とを含めてデータベース内に登録され，この登録内容から外部手続き呼び出しのコードが生成される．
- 入力項目値が値域外の場合には値域外処置により入力データを読み飛ばす（**reject**）かプログラムを異常終了させる（**abort**）．
- 属性 **time(*n*)** は一次元配列を表わし，COBOL の occurs 句に相当する．

4.3 手続き宣言

手続き宣言では，出力データと内部データとの各項目値の演算を指定する．

各項目宣言の書式：

項目名 [演算条件 1] 演算式 1
[演算条件 2] 演算式 2
・
・

演算条件 i が真ならば，項目の値として**演算式 i** の値が取られることを表わしている．同一項目について演算条件が真のものが複数個あった場合には，どの演算式の値をその項目の値とするかは**非決定**とする．この非決定性に由来する誤りを避けるために，利用者は一般には演算条件を互いに排他な条件とする．

```
注文番号 [$exist (出庫依頼)] 出庫依頼. 注文番号
品　　名 [$exist (出庫依頼)] 出庫依頼. 品名
数　　量 [$exist (出庫依頼)] 出庫依頼. 数量
送り先名 [$exist (出庫依頼)] 出庫依頼. 送り先名
```

図-5 注文記録に出庫依頼の内容を記録する部分の手続き宣言

演算条件は常に真なら **any**，さもなければ COBOL で許される条件式――条件関数を含む――である．**演算式**は単項または COBOL の COMPUTE 文で許される式――演算関数を含む――である．項目の指定は，**データ名．項目**である．同じデータ内項目については，**データ名．**を省略できる（**図-5**）．

(1) 演算部の合成方法

出力データ及び内部データの各項目値を演算する COBOL プログラムは手続き宣言から合成される．合成に際して次の事項を**前提条件**として利用している．

- 主データ項目と関連データ項目，lookuptable 属性の内部データ項目とは参照のみで代入はされない．
- 出力データ項目は，**初期値**が空であり，演算値が確定した後でその項目値が参照される．
- temporary 属性の内部データ項目については出力データ項目と同じとする．
- permanent 属性の内部データ項目の新値も，演算値が確定した後でその項目値が参照されるが，旧値を初期値とする．旧値は参照のみである．

演算部の合成方法：

手続き宣言の3つ組（代入先項目，演算条件，演算式）は COBOL の条件代入文――演算条件が any の場合には代入文――に展開される．すべての条件代入文について，その代入先項目を演算条件中または演算式中で参照している――旧値参照は除外――3つ組から展開された条件代入文の前に実行されるように条件代入文を並べ替える．

項目の参照・代入についてループがある場合には上記の並べ替えはできない．このループは合成時に検出されるが，その際にループを切る方法を説明する．提供しているアルゴリズムのひな形と上記の前提条件とにより，すべての項目値は，主データ項目値と関連データ項目値，lookuptable 属性の内部データ項目値，

permanent 属性の内部データ項目の旧値とから演算できる．たとえば，項目aを項目bから，項目bを項目cから，項目cを項目aからそれぞれ演算していたとする．この場合には項目の参照・代入について**a**←**b**←**c**←**a**のループがあるために並べ替えはできない．しかしながら，項目cの演算を項目aもbも使わない演算に必ず変えられる．そこで項目cを項目dから演算でき，さらに項目dの演算には項目aもbも使われないとする．項目cを項目dから演算するように変えれば，項目の参照・代入は**a**←**b**←**c**←**d**となりループはなくなる．項目の参照・代入にループがなければ手続き宣言で指定された通りに実行するプログラムが合成できる．

合成されるプログラムは項目値について代入以前に参照することを禁止している．さらにすべての項目を初期化してから使っている．そのために，信頼性の高いプログラムが合成される．

（2） 手続き宣言の特徴

ストラクチャード・コーディングではプログラムを順序実行と条件分岐，繰り返しとの3制御構造により表現するが，ここでは状況が異なる．

- 実行順序を意識せずに各項目ごとに値を定義する．
- 条件分岐については，次のものに制限している．

if 演算条件 then 項目:=演算式

- 繰り返しについては，配列単位での集団操作に限定している．集団操作については(4)で説明する．

（3）「在庫なしか数量不足」の判定

提供しているアルゴリズムのひな形では主データの各レコードについて同一操作が繰り返される．そのためにコンテナ在庫の各レコードを順次調べて行きながら出庫できる分は出庫して，最終的に数量不足分について在庫不足リストに出力することにする．そして，注文記録の項目数量には未出庫の注文残数を記録する．さらにレコード検査がすべて済んだか判定できるように，コンテナ在庫と更新コンテナ在庫にはファイルの最後を表わす仮想のコンテナ（**コンテナ番号 99999**）を付加する．

（4） 特殊添字 each と all

4.2 で説明した属性 time(*n*) による一次元配列については，配列と配列各要素の2つの水準がある．(1)で述べた手続き宣言3つ組の並べ替えについては配列を項目に位置付け，手続き宣言中では配列単位で記述

特殊添字 each:
品名が "name-a" であるものの数量をaの値とする．
a［品名 {each}="name-a"］ 数量 {each}
この表現はつぎの意味を表わしている．
a［品名 {1}="name-a"］ 数量 {1}
［品名 {2}="name-a"］ 数量 {2}

特殊添字 all:
品名が "name-a" のものがあったら1をaの値とする．
a［品名 {all}="name-a"］ 1
この表現はつぎの意味を表わしている．
a［品名 {1}="name-a"］ 1
［品名 {2}="name-a"］ 1
⋮

図-6 特殊添字 each と all の機能

することにする．そして，配列各要素を参照・代入するために2種類の特殊添字を導入している（**図-6**）．

- **all**：配列要素すべてに対する参照・代入に展開する．
- **each**：1つの演算中に複数の配列が記述される場合には，一般には配列間になんらかの対応がある．現在は配列の要素間に1対1対応がある場合を表現する機能を提供している．1つの3つ組中に特殊添字 each を指定された複数の配列——配列要素数は同数——が記述されている場合には，配列中での要素順序の同期を取りながら3つ組を展開する．

同一3つ組中には each と all とを混在でき，数の制限もないが，**品名 {1}** 等の each, all 以外での添字参照はできない．配列要素のためには，さらに**合計**や**最大**，**最小**等が必要であるが，現在のシステムは合計を計算する機能を提供している——**$sum（項目指定）**．

（5） 更新中の数量について

更新コンテナ在庫と出庫指示書，注文記録との項目数量の値は，コンテナ在庫の品名と数量との2項目の値により次の3つの場合に分けられる．

① 注文品名と一致しない．
② 注文品名であり，注文残数以上．
③ 注文品名であり，注文残数未満．

- **更新コンテナ在庫の項目数量**：
  ①なら コンテナ在庫の数量
  ②なら コンテナ在庫の数量－注文残数
  ③なら 0
- **出庫指示書の項目数量**：
  ②なら 注文残数
  ③なら コンテナ在庫の数量
- **注文記録の項目数量**：
  ②なら 0

```
【更新コンテナ在庫】
数量 {each}  [コンテナ在庫. 品名 {each}≠～注文記録. 品名] コンテナ在庫. 数量 {each}
             [コンテナ在庫. 品名 {each}＝～注文記録. 品名 and コンテナ在庫. 数量 {each}≧～注文記録. 数量]
                                                  コンテナ在庫. 数量 {each}－～注文記録. 数量
             [コンテナ在庫. 品名 {each}＝～注文記録. 品名 and コンテナ在庫. 数量 {each}＜～注文記録. 数量] 0
【出庫指示書】
数量         [コンテナ在庫. 品名 {each}＝～注文記録. 品名 and コンテナ在庫. 数量 {each}≧～注文記録. 数量]
                                                  ～注文記録. 数量
             [コンテナ在庫. 品名 {each}＝～注文記録. 品名 and コンテナ在庫. 数量 {each}＜～注文記録. 数量]
                                                  コンテナ在庫. 数量 {each}
【注文記録】
数量         [コンテナ在庫. 品名 {each}＝～品名 and コンテナ在庫. 数量 {each}≧～数量] 0
             [コンテナ在庫. 品名 {each}＝～品名 and コンテナ在庫. 数量 {each}＜～数量]
                                                  ～数量－コンテナ在庫. 数量 {each}
```

図-7　項目数量の数量更新用の手続き宣言

③なら　注文残数－コンテナ在庫の数量

項目数量の手続き宣言を**図-7** に示す．図中で“～”は permanent 属性内部データ項目の旧値指定である．

（6）　未確定事項

3 宣言中で未確定な事項は，プログラム宣言中の 2 つの出力条件（**出庫あり①，在庫なしか数量不足②**）と出庫指示書の項目空コンテナ搬出マークの手続き宣言③との 3 つである．

- ②については，数量不足ゆえ次の条件になる．
  コンテナ在庫．コンテナ番号＝99999
  and 注文記録．数量＞0

①と③のために，**内蔵品出庫マーク**と**内蔵品総量**との 2 項目からなる temporary 属性の内部データ**一時記録**を追加する．一時記録のデータ宣言と手続き宣言とを**図-8** に示す．これにより①と③は次のように決まる．

- ①：**一時記録．内蔵品出庫マーク＝1**
- ③：**空コンテナ搬出マーク**
  ［一時記録．内蔵品総量＝0］　1
  ［一時記録．内蔵品総量＞0］　0

（7）　手続き宣言での誤りやすい箇所

手続き宣言では，次の 2 点で間違いを起こしやすい．

- permanent 属性の内部データでの旧値と新値との使い方を誤る．
- 同一項目で複数の演算条件が排他条件でないために演算値が一意に決まらない．

```
データ宣言：
02  内蔵品出庫マーク    dec (1)
02  内蔵品総量          dec (3)
手続き宣言：
内蔵品出庫マーク
  [コンテナ在庫. 品名 {each}＝～注文記録. 品名
    and コンテナ在庫. 数量 {each}＞0
    and ～注文記録. 数量＞0] 1
内蔵品総量 [any] $sum（更新コンテナ在庫. 数量）
```

図-8　一時記録のデータ宣言と手続き宣言

注文記録の項目数量の手続き宣言（図-5，図-7）には誤りがある．出庫依頼を入力した時にはコンテナ在庫の入力レコードはないために，注文記録の項目数量には，出庫依頼の入力数量と 0 との 2 値が取り得る．条件（**$exist（コンテナ在庫）**）を加えて演算条件を排他条件にすることでこれを回避する．

課題プログラムの 3 宣言が最終的にどうなったかを付録に示す．

5. むすび

プログラム合成技術を中核としたプログラム製造システムである MOTHER SYSTEM について，課題プログラムを例に取り上げて，プログラムの設計方法とプログラム仕様に位置付けられる 3 宣言――プログラム宣言とデータ宣言，手続き宣言――とについて解説した．合成対象に選んだ事務処理プログラムの処理形態は，合成技術を開発する上で適切であった．

謝辞　システム開発は技術センターの廣道博史研究員と日本電気情報サービス(株)の藤野博之氏，斎藤明久氏，石井景子さんと日本電気技術情報システム開発(株)斎藤実氏，田口安男氏との 6 人が行った．

コンサルタント委員会委員及びワーキング委員会委員からは，多数のご意見をいただいた．

ご協力いただいた方々に深謝する．

参考文献

1) 米口：事務計算プログラマのためのプログラミング入門（2）システミング演習，bit, Vol. 12, No. 8, pp. 1032-1040 (1980).
2) 西村：事務処理業務ソフトウェアの部品分割方

式，情報処理学会ソフトウェア工学研究会資料 30-3 (1983).
3) 西村：MOTHER SYSTEM によるソフトウェアの設計と製造，情報処理学会，プログラム設計技法の実用化と発展シンポジウム論文集，pp. 103-112 (1984).
4) 西村，廣道，藤野，斎藤明久，斎藤実，田口：仕様部品からのプログラム合成システム MOTHER SYSTEMの構成，情報処理学会ソフトウェア工学研究会資料 35-3 (1984).
5) 日本電気：STEPS/C プログラミング標準，ACOS ソフトウェアエンジニアリング，YAE 12-1 (1979).

(昭和59年10月5日受付)

付録 課題プログラムの最終的なプログラム仕様

(1) **プログラム宣言**

主キー項目：コンテナ番号
主データ：コンテナ在庫
　　出庫依頼
出力データ：更新コンテナ在庫　出力条件：$exist (コンテナ在庫)
　　出庫指示書　出力条件：一時記録．内蔵品出庫マーク＝1
　　在庫不足リスト　出力条件：コンテナ在庫．コンテナ番号＝99999 and 注文記録．数量>0
内部データ：一時記録　temporary
　　注文記録　permanent

(2) **データ宣言**

【コンテナ在庫】

02 コンテナ番号	dec (5); $ 範囲 (1,99999); reject
02 搬入年月日時	
03 搬入年月	char (4)
03 搬入日時	char (4)
02 内蔵品	time (10)
03 品名	char (10)
03 数量	dec (2)

【出庫依頼】

02 コンテナ番号	dec (5); $ 範囲 (0,0); abort
02 注文番号	dec (4)
02 品名	char (10)
02 数量	dec (2); $ 範囲 (1,99); abort
02 送り先名	char (20)

【注文記録】

02 注文番号	dec (4) value (zero)
02 品名	char (10) value (space)
02 数量	dec (2) value (zero)
02 送り先名	char (20) value (space)

【一時記録】

02 内蔵品出庫マーク	dec (1)
02 内蔵品総量	dec (3)

【更新コンテナ在庫】

02 コンテナ番号	dec (5)
02 搬入年月日時	
03 搬入年月	char (4)
03 搬入日時	char (4)
02 内蔵品	time (10)
03 品名	char (10)
03 数量	dec (2)

【出庫指示書】

02 注文番号	dec (4)
02 送り先名	char (20)
02 コンテナ番号	dec (5)
02 品名	char (10)
02 数量	dec (2)
02 空コンテナ搬出マーク	dec (1)

【在庫不足リスト】

02 送り先名	char (20)
02 品名	char (10)
02 数量	dec (2)

(3) **手続き宣言**

【更新コンテナ在庫】

コンテナ番号	[any] コンテナ在庫．コンテナ番号	
搬入年月	[any] コンテナ在庫．搬入年月	
搬入日時	[any] コンテナ在庫．搬入日時	
品名 {each}	[any] コンテナ在庫．品名 {each}	
数量 {each}	[コンテナ在庫．品名 {each}≠～注文記録．品名]	コンテナ在庫．数量 {each}
	[コンテナ在庫．品名 {each}＝～注文記録．品名 and コンテナ在庫．数量 {each}≧～注文記録．数量]	コンテナ在庫．数量 {each}－～注文記録．数量
	[コンテナ在庫．品名 {each}＝～注文記録．品名 and コンテナ在庫．数量 {each}<～注文記録．数量]	0

【出庫指示書】

```
注文番号      [any] 注文記録. 注文番号
送り先名      [any] 注文記録. 送り先名
コンテナ番号  [any] 更新コンテナ在庫. コンテナ番号
品名          [any] 注文記録. 品名
数量          [コンテナ在庫. 品名 {each}＝～注文記録. 品名
                and コンテナ在庫. 数量 {each}≧～注文記録. 数量] ～注文記録. 数量
              [コンテナ在庫. 品名 {each}＝～注文記録. 品名
                and コンテナ在庫. 数量 {each}＜～注文記録. 数量] コンテナ在庫. 数量 {each}
空コンテナ搬出マーク [一時記録. 内蔵品総量＝0]                    1
                     [一時記録. 内蔵品総量＞0]                    0
```

【在庫不足リスト】

```
送り先名 [any] 注文記録. 送り先名
品名     [any] 注文記録. 品名
数量     [any] 注文記録. 数量
```

【注文記録】

```
注文番号 [$exist (出庫依頼)] 出庫依頼. 注文番号
品名     [$exist (出庫依頼)] 出庫依頼. 品名
数量     [$exist (出庫依頼)] 出庫依頼. 数量
         [$exist (コンテナ在庫)
           and コンテナ在庫. 品名 {each}＝～品名
           and コンテナ在庫. 数量 {each}≧～数量] 0
         [$exist (コンテナ在庫)
           and コンテナ在庫. 品名 {each}＝～品名
           and コンテナ在庫. 数量 {each}＜～数量] ～数量－コンテナ在庫. 数量 {each}
送り先名 [$exist (出庫依頼)] 出庫依頼. 送り先名
```

【一時記録】

```
内蔵品出庫マーク
         [コンテナ在庫. 品名 {each}＝～注文記録. 品名
           and コンテナ在庫. 数量 {each}＞0
           and ～注文記録. 数量＞0]                    1
内蔵品総量 [any] $sum (更新コンテナ在庫. 数量)
```