ソフトウェア工学　77－12
（１９９１．２．７）

プログラム部品合成システム

荻本　浩三　　下田　宏　　杉本　武和
(株)島津製作所　技術情報システム部

ソフトウェアの生産性を向上させる手法において、プログラム部品を再利用する手法が有効である。我々は、プログラム部品を自動的に合成するプログラム部品合成システムを開発した。このシステムは、データフローダイアグラムで記述(表現)されたプログラム仕様を基に、プログラム部品データベースに保管されているプログラム部品を用いて、目的とするＣ言語のソースコードを生成する。さらに、開発システムには、生成されたソースコードの実行効率や部品の再利用率を高めるための機能、及び科学技術計算用ソフトウェア開発において有効な機能がある。本稿では、プログラム部品合成システムの概要、機能、及びシステムの特徴について報告する。

Program Synthesis System

Kozo OGIMOTO, Hiroshi SHIMODA and Takekazu SUGIMOTO
ENGINEERING INFORMATION SYSTEMS DEPT. SHIMADZU CORPORATION
1 Nishinokyo－kuwabaracho, Nakagyo-ku, Kyoto 604, JAPAN.

Reusing the program components is one of the most efficient method for software productivity. We developed a program synthesis system based on this method. For synthesizing program components automatically, the system generates the source code of C language by means of program specification represented by Data Flow Diagram and reuseable program components in program database. And moreover, the system has some special functions that improve the run-time efficiency, reuseability of program components and productivity especially for scientific computing software. In this paper, we describe the outline, functions and characteristics of this system.

1．はじめに

ソフトウェアの生産性を向上させる一手法に自動プログラミングが提案されている。[1] 自動プログラミングの中でも実用性から最も期待される手法は、部品合成による自動プログラミングである。[2] 我々は、現在部品合成手法を用いたソフトウェア開発支援環境を開発している(図1)。

この開発環境においてソフトウェアの仕様を記述する手法として、構造化分析･設計手法をサポートするデータフローダイアグラム(DFD)による記述法を採用した。しかしDFDのみでは十分に設計情報を記述できないため、DFDで用いるプロセスの制御を記述するプロセスフローダイアグラム(PFD)を併用する方法を提案した。[3] さらに、DFDの記述法を一部拡張することにより、ソフトウェア仕様の記述から部品合成手法により自動的にソースコードを生成することを可能にした。

プログラム部品合成システムは、記述されたDFD、PFDなどからプログラムの制御構造を抽出し、目的とするソースコードを生成する。また、DFDにおいてプログラム部品を使用することが指定されている場合には、プログラム部品データベースよりプログラム部品を抽出して自動的に合成を行う。

さらに本システムでは、科学技術計算用ソフトウェアの開発支援も目的としているため単位(SI単位系)の取扱、行列などの演算を直接扱うことも可能である。

2．プログラム部品合成システムの概要

プログラム部品合成システムは、図2に示すようにデータフローダイアグラム(DFD)、プロセスフローダイアグラム(PFD)、データ定義(DD)で表現されたプログラム仕様、および使用するプログラム部品の情報を基に、プログラム部品データベースに保管されているプログラム部品を用いて、C言語のソースコードを生成する。

本システムではプログラム仕様からC言語のソースコードを生成する中間段階で、「プログラムモデル」という概念を導入した。プログラムモデルを採用することにより、例えば与えられるプログラム仕様の書式が変更されたり、状態遷移図などの新たな記述法に拡張された場合に、システム全体を変更する必要がなく、与えられたプログラム仕様からプログラムモデルに変換する部分(仕様変換モジュール)の変更のみでよい。また、生成するソースコードをC言語以外の他の言語に拡張する場合においても、プログラムモデルからソースコードを生成する部分(合成モジュール)の変更のみでよい。

2．1プログラム仕様

本システムの入力となるプログラム仕様は、次の3つからなる。

データフローダイアグラム(DFD)

DFDは、1枚の図面の中にデータを処理する機能(プロセス)と、その間を流れるデータフローをグ

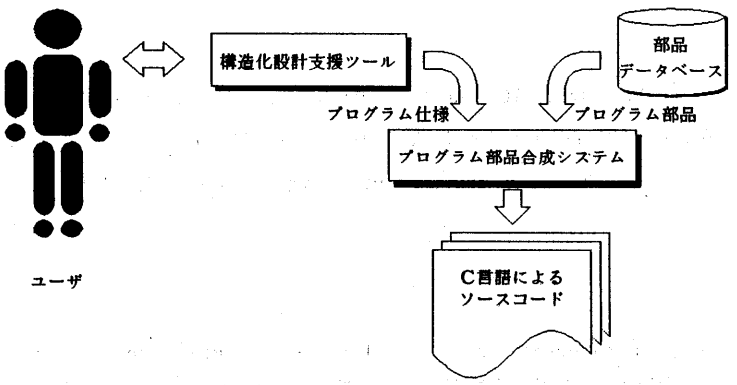


図1　ソフトウェア開発支援環境

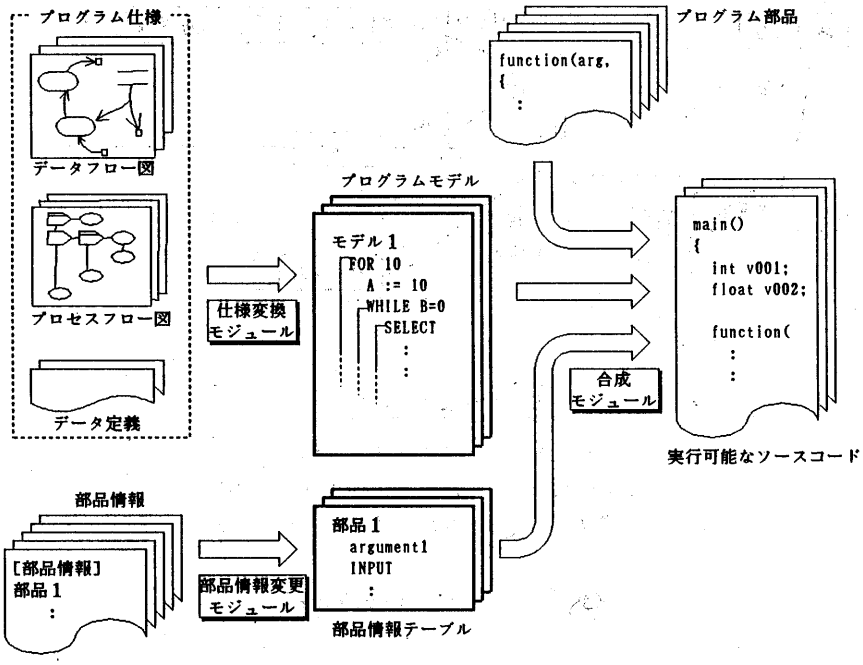


図2　プログラム部品合成システムの概要

ラフ形式で表現したものである。プロセスには、それらの機能別に異なったシンボルを当てはめ、視認性を良くしている。本システムで扱うDFDは、部品合成を行うために一般的なDFDから拡張されている。

プロセスフローダイアグラム(PFD)

DFDは、プロセス間のデータの流れを総括的に眺めたものであり、プロセスの実行順序を決定するには曖昧な部分を残している。それを補うため、プロセスの順序を直接構造化された制御図で指示したものがPFDである。

データ定義(DD)

ソフトウェアを設計する場合、そこで扱うデータがいくつかの別のデータの集まりであることが多い。このようなデータの集合を定義するものがデータ定義である。データ定義では、扱うデータ全てに名前を付けながらそれらの間の関係を定める。

2．2　部品情報

部品情報は、ソースコードを生成するためにプログラム部品データベースから検索され、合成に使用するプログラム部品に関する情報である。プログラム部品は、C言語の関数の形で記述されており、部品情報には、その関数名、引数名、引数のデータ型、データフローとの接続情報などがある。

2．3　仕様変換モジュール

仕様変換モジュールは、与えられたプログラム仕様をプログラムモデルに変換するモジュールであり、次の手順で処理を行う。

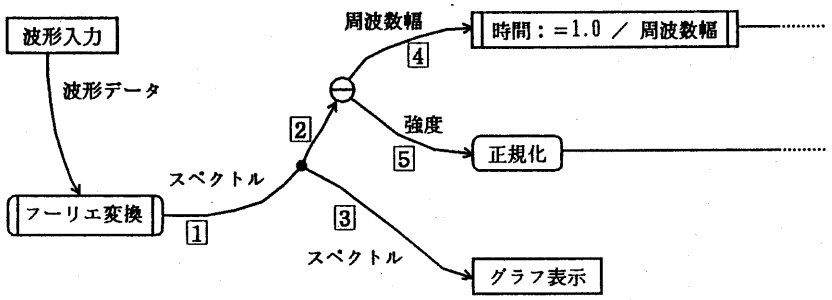
1）データフローをプログラムモデル上のデータ領域に割り当てる。

2）DFD、PFDから骨格となる制御構造を生成する。

3）DFD上の各プロセスを、モデル呼び出し、部品呼び出し、代入文等に変換する。

2．4　部品情報変更モジュール

部品情報変更モジュールは、使用するプログラム部品の情報を合成モジュールの入力となる部品情報テーブルに変更する。すなわち、与えられる部品情報において部品の引数とプログラム仕様のデータフローとの接続関係は[引数→データフロー名]で表されているのに対し、部品情報テーブルにおける接続関係は[引数→データ領域]で表す必要があるため、部品情報中のデー



データ定義（ＤＤ）

```
スペクトル ：： 周波数幅 + 強度;
周波数幅   ：： REAL <0.0..5.0E3> (Hz);
強度       ：： REAL [100];
```

図３　データ領域の共有化

タフロー名をプログラムモデルのデータ領域に変更する。

２．５　プログラムモデル

プログラムモデルはプログラムモデル内で使用するデータ（変数）を定義する「データ定義部」とプログラムの制御構造（制御フロー）を定義する「制御構造部」から構成されており、生成するソースコードの原型となる。

２．６　合成モジュール

合成モジュールは、仕様変換モジュールで作成されたプログラムモデルと使用するプログラム部品に関する情報（部品情報テーブル）からＣ言語のソースコードを生成するモジュールであり、次の手順で処理を行う。

１）合成で使用する部品の情報を整理する。
２）合成に必要なプログラム部品をプログラム部品データベースから抽出し、プログラム部品のカスタマイズを行う。
３）プログラムモデルに従って目的とするソースコードを作成する。

３．システムの機能

本システムは、 DFD などで記述されたプログラム仕様から目的とするソースコードを生成する過程で、ソースコードの実行効率や部品の再利用率を高める為に以下の機能を有している。

表１　演算式で扱える関数の例

	関数名	説明
数値計算用 （１８種類）	abs	絶対値をとる
	acos	逆余弦関数を計算する
	asin	逆正弦関数を計算する
	：	：
	：	：
ベクトル、行列計算用 （７種類）	inv	逆行列を計算する
	scl_prd	ベクトルの内積を計算する
	vct_prd	ベクトルの外積を計算する
	：	：
	：	：
文字列操作用 （５種類）	strcat	文字列を連結する
	strcmp	文字列を比較する
	strmid	文字列中の一部を抜き出す
	：	：
	：	：

３．１　データ領域の共有化

本システムでは、基本的にプログラム仕様中の１本のデータフローに付き１つのデータ領域を割り当てるが、実行時のデータ領域量やデータの移動時間等を考慮して、可能な限り共通のデー

タ領域を使用するようにしている。例えば図３のようなDFDがプログラム仕様として与えられた場合、分岐点、分解点の前後のデータ領域の共有化を行う。即ち[1]-[5]のデータフローは、本来それぞれにデータ領域が割り当てられ、５つのデータ領域を必要とするはずであるが、データ領域の共有化が行われ、1つのデータ領域を共有する。また、合成点、合流点についてもその前後で同様なデータ領域の共有化が行われる。

このようにデータ領域を共有化することにより、実行時のデータ領域および実行時間を削減し、プログラム実行時の負荷が小さくなるようなソースコードを生成する。

３．２　式変換

本システムでは、科学技術計算用ソフトウェアの開発支援も目的としているため、Ｃ言語で標準装備されている演算以外にＣ言語では直接実行できないベクトル、行列演算もサポートしている。

プログラム仕様中にはDFDに代入プロセスとして演算式を直接記述することができる。この際ベクトルや行列に関する演算式が記述された場合には、Ｃ言語でその演算機能を満たすような関数が自動的に生成される。また、その他に、表１に示すような各種の関数が用意されており、これらの関数を演算式で用いることができる。

このような式変換の機能により、プログラム仕様中にはベクトル、行列の演算をあたかも標準装備されている演算のように記述することができる。

３．３　単位変換

科学技術計算用ソフトウェアにおいては扱うデータに単位を必要とする場合がある。そこで本システムで扱うプログラム仕様中のデータについても単位が必要な場合には、単位をつけることができる。また、プログラム部品データベースに登録されている部品の引数にも単位が必要な場合には単位がつけられている。

これにより、プログラム仕様中のデータの単位と使用するプログラム部品の引数の単位が異なる場合には、自動的に単位変換を行うようなソースコードが生成される。例えば、プログラム仕様中のデータの単位がKgで接続されるプログラム部品の引数の単位がgの場合には、プログラム部品を呼び出す前にデータを1000倍するようなソースコードが生成される。この単位変換では、JISに基づく100以上の単位をサポートするために単位変換テーブルを用意している。

３．４　部品接続における型変換

本システムでは指定されたプログラム部品を接続(関数呼び出し)することにより目的とするソースコードの生成を行っている。この際意味的にはプログラム部品を接続できるが、DFD中のデータフローと引数のデータ型が異なる場合がある。

このような場合に、DFD中のデータフローと部品情報中の引数のデータ型から自動的に型変換を行うようなソースコードを生成する。

３．５　プログラム部品のカスタマイズ

プログラム部品の再利用率を高めるために目的とす

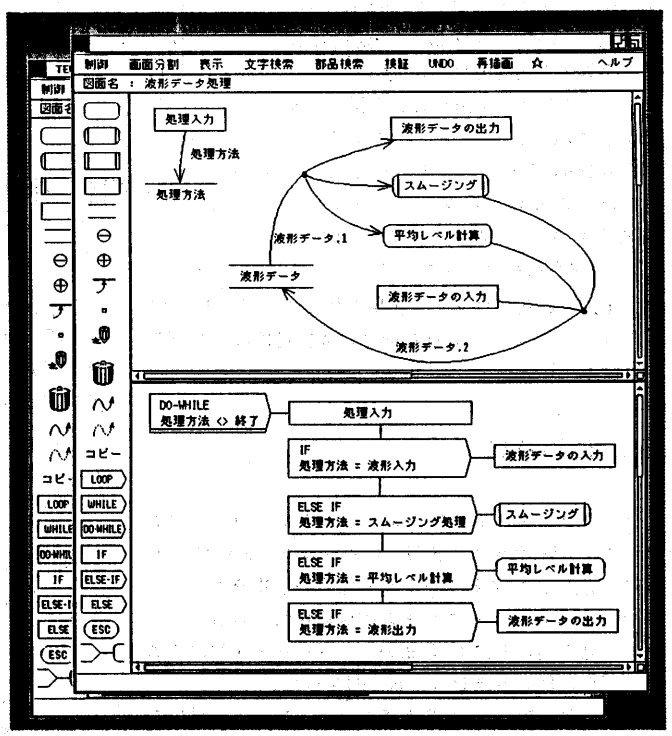


図４　プログラム仕様の記述例
（構造化設計支援ツール）

るソースコードを生成するとき、次のようにしてプログラム部品のソースコードの一部をカスタマイズする。

1）合成で使用するプログラム部品間において、名前(関数名)が同じプログラム部品が複数存在した場合に、それぞれの関数名を自動的に変更し、重複のないようにする。

2）合成するプログラム部品中に数式埋め込み部品がある場合に、その部品中にプログラム仕様で指定された数式を埋め込む。

4．システムの実行例

本システムの実行例を図4、図5に示す。図4は、構造化設計支援ツール[3]により本システムの入力となるプログラム仕様を記述しているところである。構造化設計支援ツールは、マルチウインドウ、マウスなどのユーザインターフェースによりDFD、PFD、DDなどのプログラム仕様を対話的に記述、編集するためのツールである。

一方、図5は、構造化設計支援ツールで記述されたプログラム仕様及びプログラム部品データベースより抽出したプログラム部品から本システムにより生成されたソースコードの一部である。生成されたソースコードは、カスタマイズされたプログラム部品と共にコンパイル、リンクされ、プログラム仕様通りに実行可能なプログラムとなる。

```
void    sub_function_001();     /* 平均レベル計算 */
int     input001();     /* 処理入力 */
void    input002();     /* 波形データの入力 */
void    smoothing();    /* スムージング */
void    graphic();      /* 波形データの出力 */


main()  /* 波形データ処理 */
{

    float var011[200];  /* 波形データ07 */
    long int var001;    /* 処理方法 */
    float var009[200];  /* 波形データ05 */
    float var003[200];  /* 波形データ */
    int var013;  /* NEW_TYPE013 */
    float var014[200];  /* NEW_TYPE014 */

    __assign001( var011, 0.0);
    do {
        var001 = input001();    /* 処理入力 */
        if ( var001 == 1) {
            input002(var009);   /* 波形データの入力 */
            __assign002( var011, var009);
        }
        else if ( var001 == 2) {
            __assign003( var003, var011);
            var013 = 200;
            smoothing(var003, var009, var013);  /* スムージング */
            __assign004( var011, var009);
        }
        else if ( var001 == 3) {
            __assign005( var003, var011);
            sub_function_001(var003, var009);
            __assign006( var011, var009);
        }
        else if ( var001 == 4) {
            __assign007( var00
```

図5　生成したソースコードの一部

5．おわりに

データフローダイアグラム、プロセスフローダイアグラム、データ定義で表現されたプログラム仕様、および合成で使用するプログラム部品の情報をもとに、プログラム部品データベースに保管されているプログラム部品を用いて、C言語のソースコードを生成するプログラム部品合成システムを開発した。

本システムでは、目的とするソースコードを単に生成するのみでなく、ソースコードの実行効率や部品の再利用率を高めるための機能を有している。

また、本システムならびに本システムの入力となるプログラム仕様を記述する構造化設計支援ツールを用いた実行例も併せて報告した。

今後、本システムの拡張を行い、データフローダイアグラム以外のプログラム入力仕様(状態遷移図、画面設計図、ファイル仕様書など)への対応や生成ソースコードのC言語以外(FORTRANなど)への対応を行いたい。

[参考文献]

[1] 大野ほか:自動プログラミングハンドブック、オーム社、(1989)。

[2] 古宮、原田:部品合成による自動プログラミング、情報処理、Vol28、No10(1987)。

[3] 奥山、井上、川崎、荻本:設計情報を付加した構造化仕様記述法、第41回情報処理全国大会1G-8(1990)。