SDLプログラマの提案

7M-2

田中功一, 佐藤文明, 宗森　純, 水野忠則
三菱電機(株) 情報電子研究所

1. はじめに

ソフトウェアの生産性向上が叫ばれており、解決方法がいくつか提案されている。その1つとして、プログラムの構成要素を部品化し、組合せによってシステムを構築する方法がある[1]。部品の組み合わせ方や作り方は多種多様であるが、この開発法の共通の目的として、部品の再利用による開発コストの低減化とプログラミング誤りの減少が挙げられる。

この方法は従来から事務処理用言語で利用されており、作成した部品を効率良く管理するツールも数々発表されているが、大規模複雑化する通信システムの分野へ適用する事によって同様の効果を期待出来る。

ここでは、仕様記述言語SDL[2] (Specification and Description Language)で記述された仕様書から、上記手法によって効率的にソースコードを生成することを目的としたSDLプログラマを提案する。

2. SDLプログラマ

SDLは、要求仕様を形式的に表現するための言語であり、特徴として図式表現とテキスト表現を持つ。フローチャートに似て、十数種類の箱(シンボル)を用いて仕様を記述する SDL/GRは、プロセスダイアグラムと呼ばれる図式によって処理手順を表す。

この図式を容易に取り扱うため、当社ワークステーション上には図形編集ツールSDLグラフィックエディタ(SGE)[3] が開発されており、SDLプログラマはその一部として動作する。

SDLプログラマの機能は大きく分けて次の3つの部分から構成される。

(1) 部品作成

使用する部品の定義を行なう際に、SDLプログラマは簡易エディタを提供している。部品は、シミュレーション機能のコードを付加する場合等を考慮して、1つの名前で2つの内容を定義出来る様になっている。また部品の内容はC言語を拡張した文法で記述するが、編集終了時にはUNIX上のLINTを用いた文法チェックが行なわれる。ただし、ソースコード生成の段階で、すべての部品が完成している必要はないため、進行状況に合わせた部品作成が行なえる。

(2) SDL/GRプロセスダイアグラムの組み立て

作成した部品を組み合わせてSDL/GRプロセスダイアグラムを作成する。図1に示すように部品リストに表示された項目をマウスを用いて選択し、SDL/GRプロセスダイアグラム用の編集画面に並べていく。部品リストは、シンボル毎にまとめられているので、選択を容易に行なう事が出来る。部品選択ウィンドウには、部品名の他、完成の有無、使用頻度、そしてパラメータの数を表示してある。パラメータがある部品を選択した場合、入力ウィンドウが開き、必要数のパラメータの入力を求める様になっているため、記述抜けが防止できる。

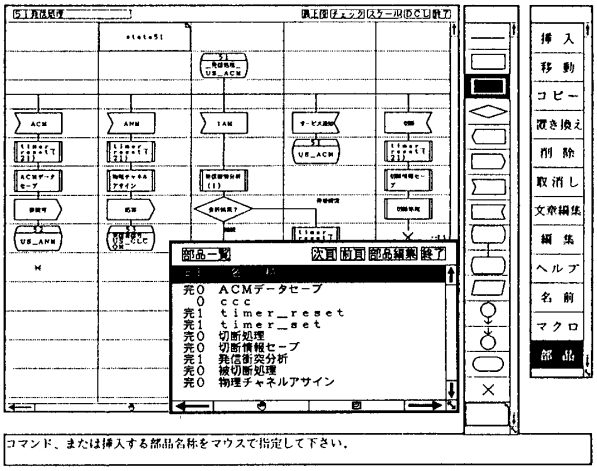


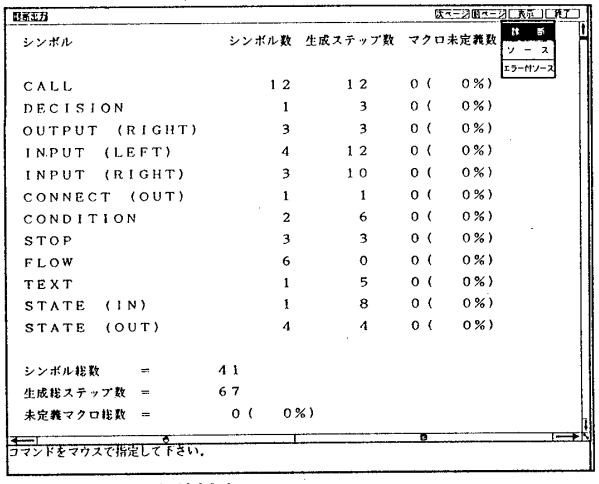
図1　　SDLプログラマ

(3) コード生成

最後に図式をソースコードジェネレータに通す事によって目的とするC言語ソースコードを生成する。生成したソースコードには部品との関連を示す情報を付加しているが、目的に応じて出力内容が設定可能である。また、部品の定義の時と同様にLINTを用いた文法チェックを行なう。SDLプログラマでは、ソースコード・プレビューア機能がありエラー情報を確認しながら図式の編集を行なうこともできる。さらに、統計情報として部品の完成度や生成コード全体に対する各部品の構成比率も出力する(図2)。

A proposal on SDL Programmer

Kouichi TANAKA, Fumiaki SATO, Jun MUNEMORI, Tadanori MIZUNO
Mitsubishi Electric Corp.



シンボル	シンボル数	生成ステップ数	マクロ未定義数
CALL	12	12	0 (0%)
DECISION	1	3	0 (0%)
OUTPUT (RIGHT)	3	3	0 (0%)
INPUT (LEFT)	4	12	0 (0%)
INPUT (RIGHT)	3	10	0 (0%)
CONNECT (OUT)	1	1	0 (0%)
CONDITION	2	6	0 (0%)
STOP	3	3	0 (0%)
FLOW	6	0	0 (0%)
TEXT	1	5	0 (0%)
STATE (IN)	1	8	0 (0%)
STATE (OUT)	4	4	0 (0%)

図2　統計情報

3. SDLプログラマにおける部品

ここで用いる部品[4]は、従来、仕様書とコーディング手順書を参考に人間が記述していたソースコードを分割したものである。その各々はSDL/GRプロセスダイアグラムで用いるシンボルにマッピングされ、SDLプログラマによって再構成される。部品は、シンボルの形と中に記述する文字列をキーとして定義する。特別な部品として、判断シンボルの様にフロー線の上に条件を記述する場合があるため、フロー線も1つのシンボルとして部品化出来るようにした。さらに、部品の内容の一部をパラメータとして与えることを可能とし、使用時にパラメータを指定できるように工夫した。

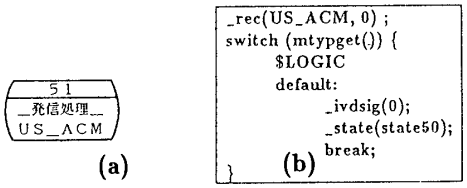


図3　状態シンボルと部品

図3(a)は、状態シンボルと呼ばれるが、これに定義された内容を図3(b)に示す。図3(b)の$LOGICで示される行は、SDL/GRプロセスダイアグラムにおいて、この部品に継る次の部品の内容が挿入される位置を示しており、入れ子構造となった部品の組み合わせ方を指示するキーワードである。また、図4(a)は9つの部品 A~Iからなる SDL/GRプロセスダイアグラムの例であり、図4(b)はそれぞれ定義した部品から生成したソースコードである。各行の左に付いている英文字は、どの部品から生成されたものかを示す。

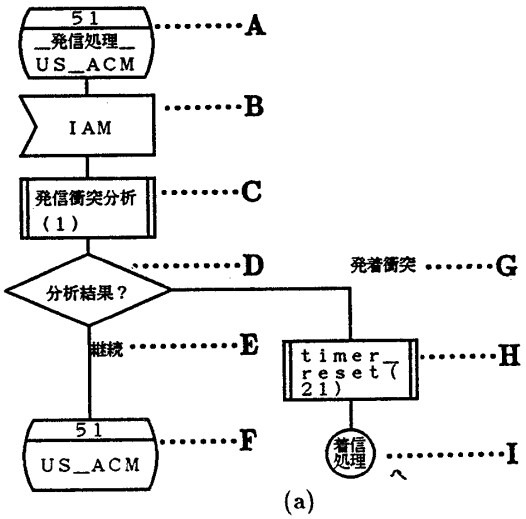
部品を用いたプログラミングは、初期の段階では部品作成の手間等も考えられるが、ある程度の部品化が進めば、処理の順序を SDL/GR プロセスダイアグラムで記述するだけで目的とするソースコードを得ることが出来るため開発の効率化が望める。

4. おわりに

従来、手作業で行なうことの多かった仕様書レベルからソースコードへのマッピングを、部品化手段を用いて半自動化した。

このように容易にソースコードが生成できる反面、デバッグを考慮した場合、ソースコードから仕様書への継りが重要な問題となる。今後は以上の点を含め、ますます重要となる通信システムを支援するために、SDLプログラマを中心とした開発環境の検討を進めて行く予定である。

参考文献
[1] 古宮他: 部品合成による自動プログラミング, 情報処理, Vol.28, No.10, pp.1329-1345 (1987).
[2] CCITT: SDL: Specification and Description Language, CCITT/Z.100 (1988).
[3] 宗森他: SDLグラフィックエディタの設計と製作, 情報処理学会論文誌, Vol.29, No.7, pp.676-685 (1988).
[4] 田中他: 状態遷移モデルから通信ソフトウェアへの写像方法についての一考察, 情報処理学会第37回全国大会 (1988).




```
A     /*51:_発信処理_US_ACM*/
A S51: _rec(US_ACM, 0);
A     switch(mtypget()) {
B         /*IAM*/
B         case MT_IAM:
C             /*発信衝突分析(1)*/
C             ret = dupcat(1);
D             /*分析結果?*/
D             switch(ret) {
E                 /*継続*/
E                 case US_CNTN:
F                     /*51:US_ACM*/
F                     _state(state51);
F                     break;
G                 /*発着衝突*/
G                 default:
H                     /*timer_reset(21)*/
H                     utimoff(UT_T21);
I                     /*着呼処理へ*/
I                     _state(state60);
I                     break;
D             }
A         default:
A             _ivdsig(0);
A             _state(state50);
A             break;
A     }
```

(b)

図4　SDL/GRプロセスダイアグラムと生成されたソースコード