モジュールの拡張について

3K－5

恐神　正博　　西田　富士夫
福井工業大学

１．まえがき

プログラムの生産性向上のためプログラムの再利用の研究が盛んであり、筆者なども基本的なモジュールや命令の呼出文のオンライン検索による設計文書の効率的な作成とこれからのプログラムへの自動変換などの研究を行ってきた。今回はもっと大きな単位で、フレキシブルにモジュールなどの再利用を図るために命令やモジュールからなる比較的大きなモジュールをフレーム的または階層的に構成し、構成モジュールや命令の選択や追加、カスタマイズなどを行う方式について検討する。

２．見出し辞書を用いた概略設計文書の作成

モジュールはプログラム作成における基本単位として重要な役割を担っているが、一つのまとまった機能と、一個の出力に限定すると、比較的小規模のサイズに限定される。また、同種であっても、詳細は異なるために、１つにまとめることが難しく、別のモジュールとして用意しなければならないことが多い。

さらに段階的詳細化を考えるとき、１つの概略処理名をいくつかの概略処理名に分割して詳細化することが行われる。一般にこのような概略処理名は元来ユーザ単位のプライベートなものであり、分かりやすく共用できる短い名前をつけることは難しい。しかし大きくてフレキシブルで再利用可能なモジュールやフレームの作成、あるいは段階的詳細化の過程に機械化を導入するため、この報告では概略処理に、一つの指標として予め名前を用意し、階層図表などにより構成を示すものとする。階層図表を用いて概略処理名は、これに属するいくつかの概略処理名やモジュール、ステートメントの呼出文を順次、視察することによりその機能を察知することができる。

ここでは基本機能を反映する簡単な呼出名をもつモジュールやフレームの構成要素の一覧を提示し、その構成要素の命令やモジュールを取捨選択しカスタマイズして仕様に沿って設計文書を作成する手法について考える。例として図１のようなファイル処理用の見出し辞書を考える。

```
dict([
 [head(ﾌｧｲﾙ読み込み処理,file_read_process),
   body([p(1,[],ﾌｧｲﾙ読み込み前処理),
         p(2,[],ﾌｧｲﾙ読み込み繰り返し処理),
         p(3,[],ﾌｧｲﾙ読み込み後処理)  ])],
        body([p(1,[],ﾌｧｲﾙ読み込み前処理1),
                ..........................  ])],
 [head(ﾌｧｲﾙ読み込み前処理1,
                        file_read_pre_process1),
   body([p(1,['PROC1','PROC2'],
     ([ﾌｧｲﾙを開く,ﾌｧｲﾙを読み込む,
      [if(終端),'PROC1',                          ①
       else,[繰り返し演算初期設定,'PROC2'],②
       end_if ]]) ])],
 [head(ﾌｧｲﾙ読み込み繰り返し処理,
                    file_read_repeat_process),
   body([p(1,SEQ_PROCESS,
      [eofまで繰り返す,SEQ_PROCESS,
              ﾌｧｲﾙを読み込む,end_repeat]),
     select([更新処理,演算処理,表示確認処理,
      検索処理,印字処理,ｺﾋﾟｰ出力処理])])], ③
 [head(ﾌｧｲﾙ修正処理,
       file_modification_process),
      body([p,(1,[PROC,PROC2],
  [更新すべきﾚｺｰﾄﾞ番号をﾏｽﾀｰのﾚｺｰﾄﾞ番号に移す,
   ﾏｽﾀﾌｧｲﾙを読む,PROC1,修正処理,PROC2])     ④
   ])]  ]).
```

図１

ユーザは所要の分野のモジュールやステートメントなどを検索するため、[1]前報告と同様に、項目headの第２引き数に書かれたその分野のキーワードKEYをadd_sp(KEY).により入力して、これに対応するbody部の引き数部を表示させ、キーワードの分野の処理のためのモジュールや命令の呼出文一覧を見る。body部はいくつかの設計文単位p(N,VAR,STATEMENT)からなる。ここに、Nは番号、

On frame-like Extention of Modules
Masahiro Osogami and Fujio Nishida
Fukui Institute of University

STATEMENTは、モジュールやステートメントなどの呼出文や見出し文、VARはSTATEMENTに含まれるカスタマイズ用変数である。③のSELECT文はbody部のカスタイマズ変数をその引き数部から選ぶことを指定する。ユーザは所要の文番号を指定し、必要に応じて変数をカスタマイズし呼出文をいくつか設計文書に加える。

3．段階的詳細化

処理p1,p2,...,pnを引き数としてもつ設計文書proc_list(p1,p2,...,pn).を詳細化するには、設計文書の中で詳細化を所望する（呼出文や見出しキーよりなる）処理piに対する処理一覧を表示させ、これから詳細化のための所望の処理の文番号を選び変数をカスタマイズしてpi1,pi2,...,pinをつくり、述語refine(pi,[pi1,pi2,...,pin]).を機械的に作る。そして設計文書の中のpiの部分をリスト[pi1,pi2,...,pin]で置き換える。これは導出法やリストの置換により機械的に行うことができる。次式はリストの要素置換のために設けた２，３の規則である。

```
rep(0,N,[O|T]):-ref_proc_list(PREC),
    append(PREC,[N],RESULT),
    ra_proc(RESULT),rep(0,N,T).          ①
rep(0,N,[X|T]):-ref_proc_list(PREC),
    append(PREC,[X],RESULT),
    ra_proc(RESULT),rep(0,N,T).          ②
rep(0,N,[]).
ra_proc(X):-retract(ref_proc_list(Y)),
    assert(ref_proc_list(X)).            ③
ref_proc_list([]).                       ④
refine_proc:-refine(0,N),
    proc_list0(PROC_LIST),rep(0,N,PROC_LIST),
    ref_proc_list(REF_PROC_LIST),
    write(REF_PROC_LIST).                ⑤
```

①②のrep(0,N,P)述語は設計文書のPの中に処理0が含まれていればこれを詳細化した処理Nに置き換えて新しく処理列RESULTを④の述語ref_proc_listの引き数部に作る。⑤のrefine_proc述語はproc_list0述語の引き数部に移した設計文書の処理列をrefine述語の引き数部で指定した処理の置き換えにより設計文書の詳細化を行うものである。

4．拡張型モジュールについて

段階的詳細化につれて具体化が必要となり、カスタマイズ用変数を含むモジュールを使って設計を進める。例えばファイルの読み込み前処理として、”ファイル,FILE,を開いて読み,VAR_LIST,を,VALUE_LIST,の値に初期設定する”というモジュールの見出し文がありこれを利用するのであれば、仮変数名を実変数名化すればよい。従来のモジュールではこのように狭い意味での変項をモジュールに含ませて汎用性をもたせており、モジュールを目的言語に変換して保存しているものが多い。しかし変項を狭い意味での変項に限定すると、モジュールの汎用性もかなり限定される。実用上現れる多くのまとまった処理の中にはいくつかの手続き文リストを変項として含ませることによりモジュール化しうるものが少なくない。このようなモジュールには多くの定型的な処理文といくつかの不定形の処理文が混在する。不定形の処理文は例えば図の①②のようにPROC1,PROC2で指定する手続き文リストの位置をモジュールの中で指定しておく。このような拡張モジュールを用いてPROCを詳細化し、通常のモジュールを作成するには、詳細化の希望分野をkeyで指定しadd_sp(key).で所望のモジュールや命令を表示選択し、PROCを具象化すればよい。不定な処理形には計算処理などのように形の上では制限のないものと、制限をもつものとがある。例えばファイルへのキーボードからの全項目の書き込みやファイルからの読み出し表示などの処理設計である。コボルやCのレコードや構造体においては同様な処理で同様な構造のものでも、項目やメンバーが異なるごとに構造や処理を明示的に定義しなくてはならない場合が多い。このような場合

```
itemlist(NAME,[[NAME1,TYPE1],
      ....[NAMEn,TYPEn]].                (1)
```

のような述語を定義しておく。そしてモジュールの中でPROCを

```
[itemlist,NAME,を登録する[または,
      定義する,表示する,更新する].  (2)
```

などの形に指定すると、(3)のような手法により変換時にモジュールのこの部分が各項目についての処理手続き文に置き換わる。

```
registr(NAME):-itemlist(NAME,ITEMLIST),
     regist_list(ITEM_LIST).
registr_list([H|T]):-registr_write(H),
     registr_list(T).
registr_list([]).                        (3)
```

registr(H)により一つの項目Hについての登録用の手続き文を生成する。

参考文献

[1]恐神正博,西田富士夫:構造化図の作成ﾁｪｯｸとｺｰﾄﾞの生成,第４７回情報処理学会全国大会