(1979. 3. 29)

階層化プログラミング言語ＳＰＬの評価

野木兼六*，中所武司*，林　利弘**，森　清三**
(日立製作所　*システム開発研究所，**大みか工場)

１．はじめに

ソフトウェア工学あるいはソフトウェア開発技術に関する研究は、１９６８年に、Dijkstraが構造的プログラミング手法[1)2)]を提案したことによって始まった。この手法は、その後、Wirthの段階的詳細化手法[3)]、Parnasの情報隠蔽によるモジュール分割手法[4)]、Liskovの抽象データ型による階層化手法[5)]などによって発展し、現在では、多くの人々の支持を得て、不動の地位を確立している。手法の発展に伴って、これを具体的に支援するプログラミング言語の開発も盛んに試みられた。まず、１９７１年に、プログラムの構造化を支援する言語として、ＰＡＳＣＡＬ[6)]が提案された。ＰＡＳＣＡＬの翻訳系は、ＰＡＳＣＡＬ自身で記述されているため、高い移行性を持ち、かなり広範囲に普及している。また、１９７４年以降、プログラムの階層化を支援する言語として、ＣＬＵ[7)8)]、ＡＬＰＨＡＲＤ[9)10)]、ＭＥＳＡ[11)]、ＥＵＣＬＩＤ[12)]などが次々と提案された。しかし、これらの言語は、実験的な色彩が濃く、まだ実用化の段階には至っていないと言って良いだろう。

我々も、制御用実時間ソフトウェアの品質向上および生産性向上を目指して、１９７５年から翌７６年にかけて、階層化プログラミング言語ＳＰＬ(Software Production Language)とその翻訳系の開発を行なった。そして、いくつかの実験プロジェクトで試用した後、１９７７年から本格的な適用を開始し、現在に至っている[13)14)15)]。

本稿では、ＳＰＬの開発や適用の過程で行なってきた各種の評価、その時に発生した問題点と対策などについて述べる。これまで、この種の言語が大規模ソフトウェアの開発に使用された例は殆んどないので、これらの情報は、その実用性や有用性を知る上で参考になろう。

２．ＳＰＬの概要

ＳＰＬの開発目標は、次の２項目に大別される。

(1) 下降型構造的プログラミング手法の実現

(2) 問題向言語開発の効率化

このような目標を達成するために、ＳＰＬは、以下の４つの特徴的な機能を持っている。

(1) モジュール設計支援機能

プログラムは、一般に、複数個のモジュール(ここでは、翻訳単位の意)から構成される。従来のプログラミング言語では、個々のモジュールを全く独立に記述することになっているので、モジュール間の構造を殆んど表現することができないだけでなく、記述の重複やインタフェースの矛盾が発生しやすい。しかも、翻訳系は、インタフェースの矛盾を発見する能力を持っていないため、開発の最終段階までこれが放置され、修正に多大の労力を要することになる。これに対して、ＳＰＬでは、機能分割に基づいて設計されたモジュール間の構造を自然に表現することができるようになっている。すなわち、ＳＰＬは、機能分割によって得られるモジュールの木構造における非端末モジュールと端末モジュールに対応

するものとして、環境モジュールと処理モジュールという２種類のモジュールを持っている。これによって、環境モジュールにおける共通データの宣言と、その子孫モジュールにおける宣言なしの使用が可能になる。また、モジュール間の構造が明確になり、インタフェースの矛盾も減少する。たとえ、矛盾が発生したとしても、翻訳系がこれを自動的に発見する。

(2) プログラムの構造化・階層化機能

構造的プログラミング手法は、プログラムの構造化と階層化から成る。階層化には、下降型と上昇型があるが、構造的プログラミング手法では、通常、前者を使用することが多く、問題向言語の開発では、後者を使用していると考えることができる。ＳＰＬでは、これらを統一的な言語体系によって支援している。このようなプログラムの階層化は、データ型宣言によるデータの階層化と手続宣言による処理手順の階層化によって実現される。この２つの階層化機能は対称性を持っており、データと処理手順の階層化を並行して進めることができる。

(3) 自然語風記述機能

階層化されたプログラムにおいては、各階層は抽象化された概念を使用して記述される。このような抽象化された概念は、設計者あるいはプログラム作成者によって、随時導入されるものであるから、その機能が直感的にわかりやすい形で表現されていないと、かえって、プログラムのドキュメント性を低下させる要因となる恐れがある。ＳＰＬでは、抽象化された概念を自然語風に記述することによって、このような問題を回避することができる。すなわち、ＳＰＬは、データ型や手続の参照形式として、次のようなものを許している。

(a) 従来形式　　ＧＥＴ（Ｍ，Ｔ）
(b) キーワード形式　ＧＥＴ（ＭＡＸ＝Ｍ，ＴＡＢＬＥ＝Ｔ）
(c) コメント形式　　ＧＥＴ　ＭＡＸ（Ｍ）　ＦＲＯＭ　ＴＡＢＬＥ（Ｔ）
(d) 混合形式　　ＧＥＴ（ＭＡＸ＝Ｍ）　ＦＲＯＭ（ＴＡＢＬＥ＝Ｔ）

(4) 目的プログラムの編集・最適化機能

階層化されたプログラムにおいては、比較的小さな手続が数多く使用され、これによる実行時の効率低下が無視できなくなる。このため、ＳＰＬでは、手続の展開形式を指定することができるようになっている。また、この時、翻訳時実行機能によって、目的プログラムの編集や最適化を行なうことができる。

ＳＰＬプログラムの例を図１に示す。

```
MODULE    LNO.    SPL  SOURCE  STATEMENTS  ( FILE INDEX H-80 SPL V2-T2 )  ( 1978  AUG. ) ID SEQ.     BLOCK NEST

          -----  -L-------1---------2---------3---------4---------5---------6---------7R--------8

                 @GENERATE
          00001   PROCESS   NYUKO (NYUKOE) ;                                                    NYU00100
          -----  -L-------1---------2---------3---------4---------5---------6---------7R--------8
NYUKO     . 00002      FUNCTION   NYUKOF ﾆｭｳｺ ｼｮﾘ ﾀｽｸ      OPT(MAIN) ;                          NYU00110
NYUKO     . 00003          RESERVE   TANAFL,ZAIKFL,TAZAFL ;                                    NYU00120
NYUKO     . 00004          CRT ﾆ ﾒｯｾｰｼﾞ'('*** < ﾆｭｳｺ ｼｮﾘﾁｭｳ > ***') ｵ ﾋｮｼﾞｽﾙ ;                  NYU00130
NYUKO     . 00005          ﾆｭｳｺ ﾊﾝﾃｲ ﾉ ｼｮﾘ  (FNYUKO) ;                                      NYU00140
NYUKO     . 00006          IF   FNYUKO .EQ. NORMAL                                            NYU00150   1
NYUKO     . 00007               THEN   ｾｯﾃｲﾊﾟﾗﾒﾝ ﾖﾘ ﾄﾘｺﾑ (RTURNB=BEND)  ;                    NYU00160   .
NYUKO     . 00008                      ﾆｭｳｺ ｼｮﾘ ｵ ｽﾙ ;                                       NYU00170   .
NYUKO     . 00009                      ﾆｭｰｺ ｼﾞｮｳﾀｲ ｵ ｼｭﾂﾘｮｸｽﾙ ;                             NYU00180   .
NYUKO     . 00010                      CRT ﾆ ﾒｯｾｰｼﾞ'('*** < ﾆｭｳｺ ｶﾝﾘﾖｳ > ***') ｵ ﾋｮｼﾞｽﾙ ;  NYU00190   .
NYUKO     . 00011               ELSE   CRT ﾆ ﾒｯｾｰｼﾞ'('*** < ﾆｭｳｺ ﾌｶﾉｳ > ***') ｵ ﾋｮｼﾞｽﾙ ;  NYU00200   .
NYUKO     . 00012          END BEND ;                                                         NYU00210   1
NYUKO     . 00013          FREE   TANAFL,TAZAFL,ZAIKFL ;                                      NYU00220
NYUKO     . 00014          STOP ;                                                             NYU00230
NYUKO     . 00015       END NYUKOF ;                                                          NYU00240
NYUKO     . 00016     TITLE '************   ﾆｭｳｺ ｵ ｵｺﾅｳ ｼｮﾘ   ************'  ;              NYU00250

          -----  -L-------1---------2---------3---------4---------5---------6---------7R--------8
```

図１　ＳＰＬプログラムの例

３．評価実験

ＳＰＬの有用性を検証するために、記述性および保守性に関する評価実験を行なった。この実験では、ＳＰＬの効果を、従来から使用されてきた拡張ＦＯＲＴＲＡＮ型高級言語ＰＣＬ（Process Control Language）との比較で測定した。ここでは、ＳＰＬの機能のうち、プログラムの構造化・階層化機能および自然語風記述機能のみを使用した。

３．１　評価実験の前提

(1) 評価実験の被実験者として、プログラム作成者を２０名選んだ。
(2) 被実験者の２０名を、事前試験によって、ＳＰＬグループ１０名、ＰＣＬグループ１０名の２グループに分けた。
(3) 被実験者の２０名についてみると、ＰＣＬの経験は２～３年のベテランで、ＳＰＬの経験は全くない。
(4) ＳＰＬの教育は、毎日２時間で５日間、合計１０時間行なった。また、前日に出した問題の解答をＳＰＬ中心に行なった。

３．２　評価実験の方法

(1) 被実験者２０名のグループ分け
〔目的〕プログラム作成能力が同程度になるようにグループ分けする。
〔方法〕被実験者２０名に、ＧＦＣ（General Flow Chart）なしの問題を与え、ＰＣＬでプログラムを記述させて、その得点により、次のようにグループ分けする。すなわち、４を法として０番目と３番目の人をＰＣＬグループに、１番目と２番目の人をＳＰＬグループに入れる。
〔問題〕与えられたテーブルの内容を大小順に並べかえる。

(2) ＳＰＬの記述性の測定
〔目的〕ＳＰＬの記述性を対ＰＣＬ比で測定する。
〔方法〕機能仕様書、ＧＦＣ、テーブル仕様書を与えて、単位時間内でプログラムを記述させ、各グループの得点の平均点を測定する。
〔問題１〕Ｎ人の得点が格納されているテーブルをもとに、得点の平均点が同一になるようにグループ分けする。
〔問題２〕氏名と電話番号をＣ／Ｒから入力し、アルファベット順に氏名と電話番号をＬ／Ｐに出力する。
〔問題３〕部品倉庫から、部品の入出庫を行なう。

(3) ＳＰＬの「不良発見のしやすさ」の測定
〔目的〕ＳＰＬの「不良発見のしやすさ」を対ＰＣＬ比で測定する。
〔方法〕機能仕様書、ＧＦＣ、テーブル仕様書と、不良（bug）を複数個内在させたプログラム・リストを与えて、単位時間内に不良を発見・修正させ、各グループの得点の平均点を測定する。
〔問題〕部品組み立て工場において、ベルト・コンベアを制御する。

(4) ＳＰＬの「機能理解のしやすさ」の測定
〔目的〕ＳＰＬの「機能理解のしやすさ」を対ＰＣＬ比で測定する。
〔方法〕概略仕様書、テーブル仕様書、プログラム・リストを与えて、詳細機能を判読させ（質問に答えさせたり、ＧＦＣを完成させたりして）、各グループの得点の平均点を測定する。
〔問題〕機械加工工場において、加工の進度を管理する。

表1　評価実験の結果

評価大項目	評価小項目	方法	解答時間	SPLグループの平均点	PCLグループの平均点	SPL/PCL
記述性	記述性 ♯1問	機能仕様書、GFC、テーブル仕様書を与えてプログラムを記述させる	2h	58.3	76.4	0.76
	〃 ♯2問	〃	2	71.0	80.0	0.89
	〃 ♯3問	〃	4	77.5	85.0	0.91
保守性	不良発見のしやすさ	機能仕様書、GFC、テーブル仕様書、プログラム・リストを与えて、不良を発見・修正させる。	2	85.0	80.0	1.06
	機能理解のしやすさ	概略仕様書、テーブル仕様書、プログラム・リストを与えて、詳細機能を判読させる。	2	60.0	54.6	1.10
	機能変更のしやすさ	1度理解させたプログラムの機能変更を指示し、プログラムを変更させる。	2	79.6	64.0	1.24

(5) ＳＰＬの「機能変更のしやすさ」の測定

〔目的〕ＳＰＬの「機能変更のしやすさ」を対ＰＣＬ比で測定する。

〔方法〕(4)で使用したプログラムを説明し、内容を理解させた後に、機能の変更点（テーブルとその操作ルーチンの変更）を指示し、プログラムを変更させ、各グループの得点の平均点を測定する。

〔問題〕(4)と同じ。

3．3　評価実験の結果と考察

評価実験の結果を表1に示す。記述性については、3回実験したが、いずれもＳＰＬの方がＰＣＬよりも悪かった。それまでＦＯＲＴＲＡＮ系の言語に親しんでいたプログラム作成者にとって、ＳＰＬのようなＡＬＧＯＬ系の言語は、かなり抵抗があったようである。しかし、実験の回数を追うごとにＳＰＬの成績が良くなっており、ＳＰＬの習熟時には、ＰＣＬと同程度あるいはそれ以上になるものと思われる。保守性については、「不良発見のしやすさ」、「機能理解のしやすさ」、「機能変更のしやすさ」の全ての実験において、ＳＰＬの方がＰＣＬよりも優れている。これは、自然語風記述機能によるドキュメント性の向上やプログラムの構造化・階層化機能による局所性の向上などによるものと思われる。ＳＰＬの習熟時には、さらに大きな効果が期待できよう。

4．プロジェクトへの適用

ＳＰＬの有用性は、最終的には、プロジェクトへの適用によって検証されるべきであろう。ここでは、ＳＰＬの最初の適用プロジェクトを例にとって、適用時に発生した問題点や適用の効果について述べる。このプロジェクトで開発したプログラムの規模は約１３０Ｋ語（１語は１６ビット）であり、このうち４０Ｋ語に対してＳＰＬが使用され、残りの９０Ｋ語に対してＰＣＬが使用された。

まず、適用時に発生した言語上の問題点として、以下のものが指摘された。

(1) 定数式

汎用的なテーブルを設計しようとする場合、配列体の上下限などに、定数宣言で宣言された定数名を含むような定数式を書きたくなる。特に、引数つきデータ型宣言によって、このようなことを行なうには、定数名として引数名を許す必要

がある。ＳＰＬにはこの機能がなかったために、引数つきデータ型宣言の機能がうまく活用できない場合があった。

(2) 外部手続の名称

プログラムを自然語風に記述するために、ＳＰＬでは、手続の参照形式の１つとして、コメント形式を許している。コメント形式は、一般に、識別子の並びと引数の位置によって認識されるが、外部手続の場合には、最初の識別子以外は無視され、しかも、最初の識別子の長さは６文字以内に制限される。このような問題は、手続に対する外部名の指定を許すことによって解決される。

(3) 仮引数の参照

プログラムを比較的小さな手続によって階層化する場合には、ある手続の仮引数が、その手続内では使用されないで、他の手続にそのまま渡されるという事態が頻繁に発生する。これは、手続間のインタフェースを明確にするという意味で重要なことであるが、いちいち引数として渡すのは面倒な場合も多く、目的プログラムの効率もその分だけ低下する。

(4) 引数の型の検証

ＳＰＬでは、引数の型の検証を厳しく行なっているため、システム・プログラムを記述したり、既存のプログラムと連結したりするような場合に、若干不便を感じることがある。このような問題は、仮引数の型としてＡＮＹ型のようなものを許すことによって解決される。

一方、ＳＰＬ適用の定性的効果としては、以下のものがあげられている。

(1) ＳＰＬでは、共通データの設計が完了しないとプログラムの翻訳ができないので、事前に十分な設計が行なわれ、後戻り作業が少なくなる。

(2) ドキュメント性が向上し、机上検証が楽になる。

(3) 翻訳系によるインタフェースの検証が厳しく、不良が前倒しに出る。

(4) 共通データの一元管理機能やインタフェースの検証機能により、仕様変更がもれなく全員に徹底される。

また、ＳＰＬ適用の定量的効果としては、不良の混入防止と不良の早期発見に関する評価データを収集した。不良の混入防止については、ＳＰＬを使用した場合のプログラミング不良発生率は、ＰＣＬを使用した場合と比較して、５９％に減少するという結果が得られた。不良の早期発見については、図２に示すように、センタ・テストでの不良発見率が、ＰＣＬを使用した場合には５６％であったものが、ＳＰＬを使用した場合には８０％に向上し、実機テストまで残っている不良が大幅に減少するという結果が得られた。

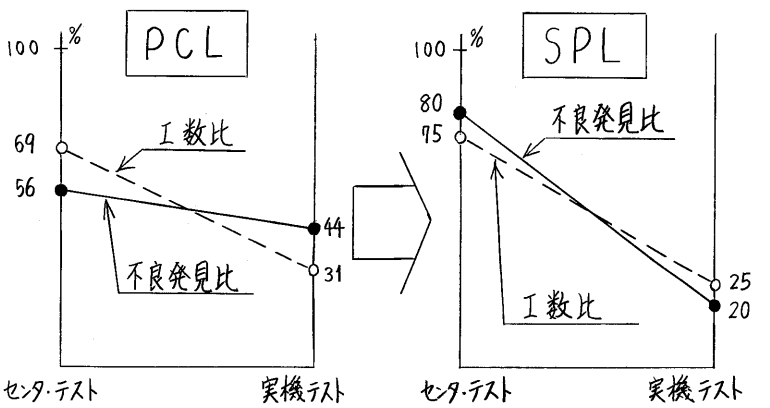

図２　不良の早期発見

5．おわりに

以上、ＳＰＬの評価について述べた。ここで得られた結果は、まだ非常に不十分なものであり、今後も、各種の評価データを着実に蓄積していく必要がある。

最後に、本研究にご援助頂いた、日立製作所大みか工場の宅間豊副工場長、北之園英博部長、高井矢庫副技師長、ならびに評価実験にご協力頂いた、ＨＥＣの加藤木和夫氏に感謝の意を表します。

参考文献

1) Dijkstra, E.W.: The Structure of the "THE"—Multiprogramming System, CACM, Vol. 11, No.5 (1968).
2) Dijkstra, E.W.: Notes on Structured Programming, Structured Programming, Academic Press, pp 1-82 (1972).
3) Wirth, N.: Program Development by Stepwise Refinement, CACM, Vol. 14, No. 4, pp 221-227 (1971).
4) Parnas, D.L.: On the Criteria to be used in Decomposing Systems into Modules, CACM, Vol. 5, No. 12, pp 1053-1058 (1972).
5) Liskov, B.H.: A Design Methodology for Reliable Software Systems, FJCC-72, pp 191-199 (1972).
6) Wirth, N.: The Programming Language Pascal, Acta Informatica 1, 1, pp 35-63 (1971).
7) Liskov, B.H. and Zilles, S.N.: Programming with Abstract Data Types, SIGPLAN Notices, pp 50-59 (1974).
8) Liskov, B.H and Snyder, A.: Abstraction Mechanisms in CLU, CACM, Vol. 20, No. 8, pp 564-576 (1977).
9) Wulf, W.A.: Alphard: Toward a Language to Support Structured Programs, Internal Report, Carnegia-Mellon University, Pittsburgh, April (1974).
10) Shaw, M., Wulf, W.A. and London, R.L.: Abstraction and Verification in Alphard: Defining and Specifying Iteration and Generators, CACM, Vol. 20, No. 8, pp 553-564 (1977).
11) Geschke, C.M., Morris Jr., J.H. and Satterwaite, E.H.: Early Experience with Mesa, CACM, Vol. 20, No. 8, pp 540-553 (1977).
12) Lampson, B.W. et al.: Report on the Programming Language Euclid, SIGPLAN Notices, Vol. 12, No. 2 (1977).
13) 林，野木，中所，浜田　他：制御用ストラクチャード・プログラミング言語ＳＰＬの開発思想　他，情報処理学会第１７回全国大会講演論文集，No. 1~4 (1976).
14) 野木　他：トップダウン・プログラミング言語ＳＰＬにおけるモジュール概念について，ソフトウェア工学研究会資料３－１（1977).
15) 林　他：制御用トップダウン・ストラクチャードプログラミング言語 —ＳＰＬ—，日立評論，Vol. 60, No. 3, pp 59-64 (1978).