ソフトウェア工学における日本語の役割

神田泰典　　杉本正勝
富士通株式会社

1. はじめに

「わが国のコンピュータは，ハードウェアは国際的にトップクラスにあるが，ソフトウェアはまだ駄目だという識者がいる。・・・わが国のソフトウェアの立遅れを，コンピュータ・メーカの技術開発力の弱さとさめっけた論も，それだけ優れたハードウェアが作れる企業が，ソフトウェアだけは別だとするわけで，妙な話である。何かソフトウェアには，わが国の国民性に適さない神秘的な何かがあるとでもしないと論理が合わない。

これだけ国民全体として教育が普及していて，勤勉な国民に，それもおいしいから，今度は鉛先は教育に向けられて，・・・」という論文[1]を最近目にすることがあった。

わが国のソフトウェアの設計／作成法のレベルを飛躍的に向上させる基本的な解決方法の一つをこれから述べて行こう。

2. ソフトウェア工学における日本語の役割

日本人は通常，日本語を用いて生活上のコミュニケーションを行なっている。また，複雑な技術情報も主に日本語を主体とした技術文書の形式を用いている。プログラムを含む計算機の利用技術としてのソフトウェアは高次の概念，複雑な処理アルゴリズムについての思考を必要とする。日本人のソフトウェア設計者／作成者が思考し，思考の結果を他人に見せられるように記述するには日本語を用いて行なうのが最も自然である。この論文の主旨は，ソフトウェアのライフサイクル（システムの計画／設計，プログラムの設計／作成／検査／保守）を通じて日本語処理を導入して，ソフトウェアの設計／作成法のレベルを飛躍的に向上させようということである。

日本語処理とは，「計算機に日本語文章を入力し，計算機で所定の処理を行ない，処理の結果にもとずき，計算機が日本語文章を出力する処理」と理解していただきたい。

日／米のソフトウェア・ギャップは増大している。対策は？

米国では，ソフトウェアエンジニアリングの体系のもとでIPT等の手法で多くの問題点が解決している。米国ではソフトウェアの関係者の間で，ソフトウェアの生産物であるプログラム・リスティングを見ることで他人が設計／作成したプログラムが理解出来るレベルになっていると推定される。

IBM社FSDのMills氏は論文[2]の内で，「次世代のプログラマは，自分の作成するプログラムのエラーの数を少なく出来るであろう。多分，1年間に1個ぐらりのエラーの発生の程度になり，プログラマはプログラマとして仕事を行なっている期間に彼がおかしたエラーの1つ1つをはっきり記憶してりられるだろう」と述べてります。

この技術レベルは，「Top Down / Structured Programmingの手法と，ソフトウェアの完成前にプログラムを複数の開発担当者によってチェックすること，及びプログラムの開発担当者以外の人にプログラムのチェックを依頼することで達成出来る」としてります。プログラマがおかす年間のエラー数の予想には楽観すぎる面もありますが，とにかくこの方向を示すものに例えば，IBM社の"An Introduction to Structured Programming in PL/1"[3]のマニュアルには次のような記述がありす。

「データまたはラベルに対する名前をつけるときには，そのプログラムの読者がプログラムの機能と構造について出来るだけ理解しやすりように十分考慮してつけなさり。　・・・

経験によれば，うまく構造化されたPL/1のプログラムは，大体において，うまくその性質を示すデータ名またはパラグラフ名を用りれば，そのプログラム自身を十分説明するもの（Self Documenting）になることが判ってります。コメントは使用しないでプログラムを作るようにしなさり。もしコメントを使用する場合でもプログラムの読み易さをそこなわなりように構成し，フォーマットを定めなさり。　・・・　」

例として次のプログラムがのってります。

```
          ・・・
FINAL_TOTAL = 0 ;
DO WHILE ( THERE_IS_MORE_SALES_DATA ) ;
   DISTRICT_TOTAL = 0 ;
   PREVIOUS_DISTRICT = DISTRICT ;
      DO WHILE (( DISTRICT = PREVIOUS_DISTRICT ) &
                THERE_IS_MORE_SALES_DATA ) ;

         SALESMAN_TOTAL = 0 ;
         PREVIOUS_SALESMAN = SALESMAN ;
         DO WHILE (( DISTRICT = PREVIOUS_DISTRICT ) &
                   ( SALESMAN = PREVIOUS_SALESMAN ) &
                   THERE_IS_MORE_SALES_DATA ) ;
            SALESMAN_TOTAL = SALESMAN_TOTAL + SALES_DOLLARS ;
            GET FILE ( SYSIN ) EDIT
                  ( SALESMAN , DISTRICT , SALES_DOLLARS )
                  ( SKIP , F(5) , F(3) , F(7,2) ) ;
         END ;
               ・・・
```

以上のプログラムは，PL/1 プログラミング言語の知識と簡単な英語の理解力があれば，その1つ1つのステートメントの意味まで分かり，プログラムの作成者以外でも「なるほど，このプログラムは与えられた仕事をするプログラムであり，エラーはなさそうだな！」という結論に達するであろう。このように英語の文章形式でプログラムを作ることで，米国人はプログラム・リスティングをプログラムの作成者個人だけの理解できる対象から，プログラム作成者以外の人が理解できる対象へと質的な変化をさせて来た訳である。

さて，日本人と上記のPL/1 プログラムの例との関係はどうか？

(1) プログラムの作成工程
よほど英語に堪能な人でないと，上記のプログラムで使用してりるようなデータ名を用いて米国人のスピードでなプログラミング出来ない。

(2) プログラムの検査・保守工程（他人が作成したプログラムを理解しようとするとき） 日常英語を利用してりる米国人の担当者の方が作業の効率（速度，信頼度）はよくなる。

日本人はカナ文字またはローマ字を使った日本語文章を使用したらよいではないかという疑問にっいては，カナ文字またはローマ字の日本語文章は日常使用してりる漢字カナ混り文に比して読みづらいという欠点がある。

以上のように，米国でよりよいソフトウェアの設計／作成技術が開発され，米国に於て効果が発揮されても，日本のソフトウェア関係者には直接同じ量の効果をもたらさず，かなり割り引いた恩恵を受ける結果になってりると考えざるを得ない。特に，ソフトウェアを抽象的に（大ずかみに）とらえてそれを記述し，段階的に詳細化して行く Top Down / Stepwise refinement の手法には，漢字カナ混りの日本語文章が使用出来る日本人用のツールが必須になってりる。

米国のプログラマが，彼または彼女のプログラムを理解し易いように，以下に示すコメントを入れるのは容易であろう。一方，日本人のプログラマは日本語→英語翻訳の作業をともなり。かつ，日本人のソフトウェア関係者が次にこのコメントを読むときには英語→日本語翻訳の作業が必要となる。2回の翻訳作業が入れば，情報のコミュニケーションの行違いの可能性も大になってくる。日本人が日本語で書く技術文書でも著者が書いたようには理解してもらえないことがしばしばあるのである。

```
/*  THIS  ROUTINE  RECEIVES  CONTROL  TO  FORMAT  THE  SECOND          */
/*  LINE  OF  OUTPUT  FOR  CONTROL  LAYER  RECORDS .  ON  ENTRY , TEXTLN AND */
/*  TEXTADDR  ARE  ALREADY  SET  TO  THE  CORRECT  VALUES  TO  CALL     */
/*  SUBROUTINE  FRMTEXT .  FIRST ,  THE  DNODE  LABEL  AND  DATA  IS  MOVED  */
/*  TO  THE  BUFFER .                                                   */
```

以上のことをまとめて図式化すると図1のようになる。

ソフトウェアは人間の意識の反映である。この意識を他人に知らせるには使用しやすいことばを利用するのが当然である。また，ことばは単独では存在せず，そのことばを使用している人間の集りの文化を背負っている。このことばより，少ない量の記述でも正確なコミュニケーションが達せられるのである。

Iに示すように，米国人には，英語と米国文化の内にいるソフトウェア関係者の意識の「全面的な記述手段」があるのに対して，日本人のソフトウェア関係者は英語力という狭いチャネルを通して米国のソフトウェアの土俵に近づこうと努力しているにすぎないように思う。

将来，日本人はIIに示すように日本人の意識の反映が直接的になるようにしなくてはならない。

当然，ソフトウェアの国際互換性の問題が生じるが，この問題は計算機を使用した自動変換の手法で，好ましい解決法があると確信している。

3. 日本人のソフトウェア設計／作成法とツール

この議論を通して次のような，日本人のソフトウェア設計／作成法が結論づけられる。

(i) トップダウン／階層的詳細化

日本語の文章を自由に用いてソフトウェアの機能を抽象的に記述し，その記述を順次詳細化して行く

(ii) プログラム・リスティングのドキュメント性を増大

階層的詳細化の最終の段階の記述としてのプログラム・リスティングで，プログラムの機能が読みとり易いように，日本語文章を主体とした表現を採用する。

また，計算機を用いたツールとしては次の2個がまず必要になる。

(i) 日本語文章を用いたトップダウン／階層的詳細化を補助するツール

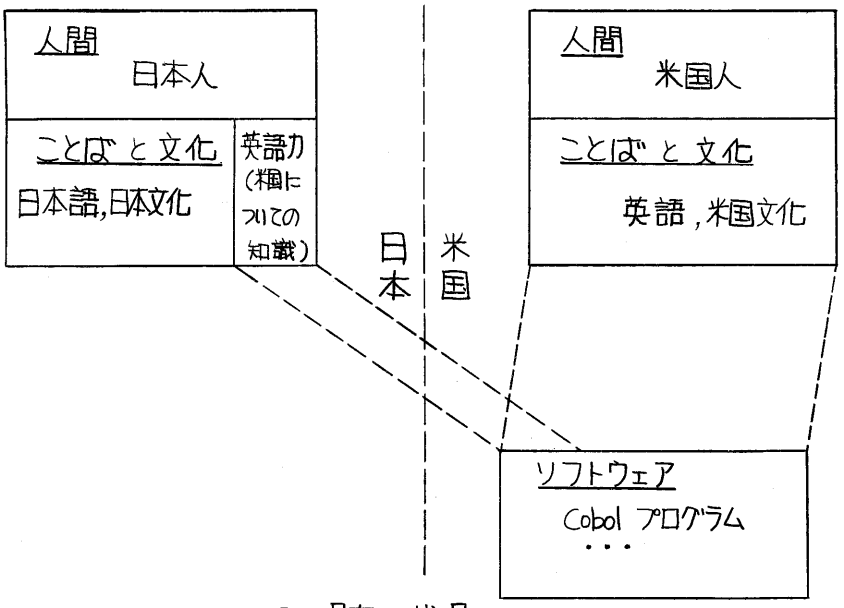
このツールは計算機と対話形式で利用出来ることが望ましい。

(ii) 日本語プログラミング言語

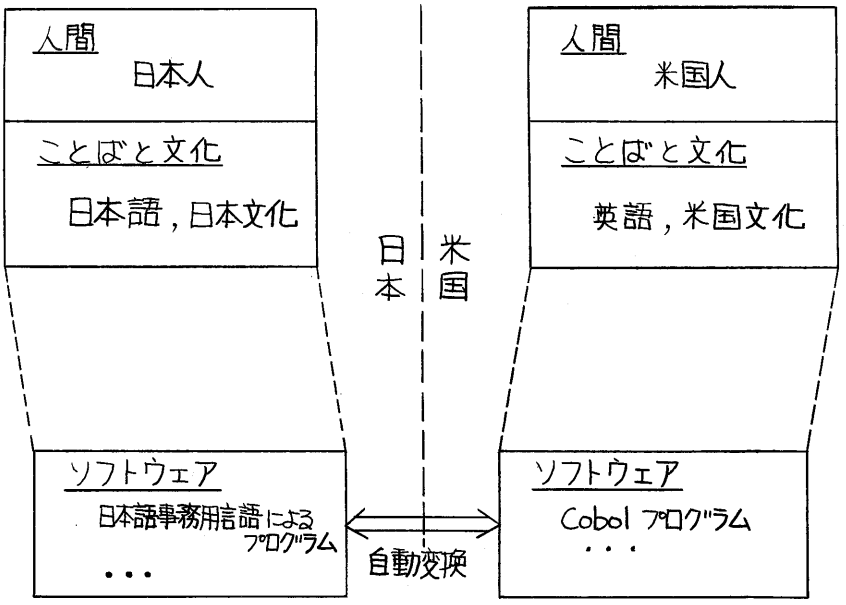
日本のソフトウェア関係者がプログラムを読んで理解し易いことを主眼的に考慮した日本語プログラミング言語が必要です。この言語は，COBOL，PL/I，APL，FORTRAN，PASCALというように既存の言語が専門分野ごとに使い分けられているのと同様に，専門分野ごとに必要となると考えます。

図2は，事務用の日本語プログラミング言語で書いたプログラムがどのようなイメージになるかを示したものです。記述はCOBOL言語に沿って，JIS COBOLの日本語用語にのっとって書いたつもりです。

実際に，日本語プログラミング言語の制定のためには国際的な動向を見ながら，国内の有識者の方々の討論と，しかるべき標準化組織の承認が必要で



I. 現在の状況



II. 将来の方向

図1. 人間とソフトウェア

す。　早い時期に日英語プログラミング言語の確立を望む次第であります。

・・・

入出力節.
ファイル管理.
　　売上ヴーＰＴ　を選択し　ＰＴ－１　へ割当てる.
　　売上ヴーＭＴ　を選択し　ＭＴ－１　へ割当てる.
　　マスタ　を選択し　ＤＰ－１　へ割当てる
　　　乱呼出し
　　　指標付き編成
　　　記号キーは　商品コードキー.
データ部.
ファイル節.
ファイル記述　売上ヴーＰＴ
　　ラベルレコードは　なし
　　データレコードは　ＰＴ－領域.
０１　ＰＴ－領域.
　　０２　区分　　形式　９９.
　　０２　得意先コード　形式　９(４).
　　０２　商品コード　形式　９(１０).
　　０２　数量　　形式　９(５).
　　０２　単価　　形式　９(７).
　　０２　金額　　形式　９(７).
ファイル記述　売上ヴーＭＴ
　　ブロックの大きさは　２０　レコード
　　ラベルレコードは　標準
　　ＩＤの値は　'売上ヴーファイル'
　　データレコードは　ＭＴ－領域.
０１　ＭＴ－領域
　　０２　区分　　形式　９９.
　　０２　得意先コード　形式　９(４).
　　０２　商品コード　形式　９(１０).
　　０２　商品名　　形式　Ｘ(１５).
　　０２　数量　　形式　９(５).
　　０２　単価　　形式　９(７)
　　０２　金額　　形式　９(７).
　　０２　仕入単価　　形式　９(７).
ファイル記述　マスタ
　　ブロックの大きさは　３０　レコード
　　ラベルレコードは　標準
　　ＩＤの値は　'マスタ'
　　データレコードは　ＤＰ－領域.
０１　ＤＰ－領域
　　０２　商品コードキー　形式　９(１０)

```
        02  商品名          形式  X(15).
        02  在庫            形式  9(7).
        02  仕入単価        形式  9(7).
        02  仕入数          形式  9(8).
        02  売上数量        形式  9(8).
手続き部.
始め.   入力として 売上ヴーPT 出力として 売上ヴーMT
        入出力として マスタ を開けよ.
読込み.
        売上ヴーPT を読め 終りでは 終り に行け.
        PT領域 の 商品コード を 商品コードキー
        に転記せよ.
        マスタ を読め 無効キーの場合 エラー処理 に行け.
        DP-領域 を MT-領域 に対応をとって転記せよ.
        PT-領域 を MT-領域 に対応をとって転記せよ.
        売上数量 に PT-領域 の 数量 を加えよ.
        PT-領域 の 数量 を 在庫 から減じよ.
        DP-領域 を再書込みせよ.
        MT-領域 を書け.
        読込み へ行け.
エラー処理.
        ' 該当キーがない' PT-領域 の 商品コード を
        表示せよ.
        読込み へ行け.
終り.   売上ヴーPT 売上ヴーMT マスタ を閉じよ.
        ' 終り ' を表示せよ.
        実行停止
        プログラム の終り.
```

図2. 事務用日本語プログラミング言語のイメージ

4. 日英語処理をサポートするハードウェアとソフトウェア

3で述べたようなツールをサポートするハードウェアの詳細は紙面の都合から述べれませんが，日英語文章の入力，日英語文章の処理，日英語文章の出力のいずれについても，近い将来に広く使用出来るようになります。

ハードウェアそのものから言えば，LSI化されたプロセッサ，LSI化された高密度メモリ，バブルメモリ等着実に開発が行なわれ，超LSIの開発にも，十分期待が出来る訳です。(4),(5)

また，日英語処理をサポートするソフトウェアは，ソフトウェア開発者の新しい開発目標になります。

5. おわりに

「・・・ さらにもう一歩あと，1976年には米国政府に対してある勧告案が提出されましたが，それは「歴史的な偶然によって米国は，ソフトウェア作りの現行の進めかたに関して，世界の他の部分よりも多くの慣性（および既得权）を持ってりる」という。彼らにとって心配な状況があることに対して注目を求めてります。この報告書は米国が，ソフトウェア開発の進め方という面で，他のもっと柔軟性をもった国民にとって代られるおそれがある，と指摘してりるのです。」という記述を最近「情報処理」[6]で読みましたが，

日英語は，漢字カナ混り文を使用し，読み易さにおりて英語より優れてりる言語とも考えられます。更に日本語の文章の中には英語のword，ギリシャ文字等が自然の形で導入出来ます。それ故，英語の文章と比較してよりよいコミュニケーションの手段となり得ましょう。そこで，この日英語を「計算机」を使用して入力，処理，出力する」環境を整備し，米国のソフトウェアに対し（とって代るまで行かなくとも），十分競争力を持つソフトウェアを設計／作成するための基盤が出来ると確信します。

また，日英語を主体としたソフトウェアの設計／作成によって，ソフトウェアを設計／作成出来る人の年令層が大きく広がることも注目すべきだと考えます。

謝辞

いつも御指導いただいてりる開発事業部 山田部長，方式部 井上部長および，討論いただいた沢井氏に 深謝いたします。

~~参考資料~~

1. ビジネス・コミュニケーション　Vol. 15　No. 1　1978
「コンピュータ・メーカへの期待」　Page 43.
西野博二著

2. in Current Trends in Programming Methodology, 1977, Prentice-Hall
"On the Development of Large Reliable Programs"
by R. C. Linger, H. D. Mills

3. IBM June 1977
"An Introduction to Structured Programming in PL/1"
GC20-1777-1

4. 昭和52年度　電子通信学会　情報・半導体部門全国大会特別講演
「超LSI技術とコンピュータの将来」
小林大祐著

5. 情報処理学会　計算機アーキテクチャ研究会　1977.11.29
「コンピュータにおける産業革命」
神田泰典著

6. 情報処理　Vol. 18　No. 12　1977
「プログラミング―工芸から科学へ」　Page 1255.
E. W. ダイクストラ著,　木村泉　和田英一　共訳