ソフトウェア工学 40－8
（１９８５．２．７）

日本語文章推敲支援ツールのプロトタイピング

牛島和夫，日並順二，尹志熙，高木利久
（九州大学工学部）

１．はじめに

ソフトウェアの開発に当たって，従来のライフサイクルモデルへの反省から，プロトタイピングの手法が最近注目されている［１］．

日本語文章推敲支援ツール「推敲」［２］は機械可読な形で存在する日本語文章を解析してその推敲に役立つ情報を提供することを目的としたシステムである．その開発に際して，仕様をあらかじめ明確に与えることができないため，プロトタイピングの手法を採用することが相応しい［３，４，５］．

プロトタイプは使い捨て型と改良発展型の二種類に大別できる［１］．どちらかといえば，前者は従来の要求仕様定義の代わりに用いられることが多く，後者は研究的色彩が強いソフトウェアの開発に利用されることが多い．「推敲」におけるプロトタイピングは仕様の確定という観点からは前者とみなせるが結果的には後者に属するものであり，徐々に改良して実用システムに近づけるという方式によりシステムを開発した．このような手法を使ってソフトウェアの設計や開発を行った事例は既にいくつか報告されているけれども［１］，今回，我々が行った「推敲」のプロトタイピングでは従来あまり議論されなかった側面が現われたので，ここに紹介する．

本報告ではプロトタイピングの観点から「推敲」の開発経過を順を追って述べることにする．

２．「推敲」

２．１　開発動機

日本語ワードプロセッサの普及により，文章の作成が容易になってきた．これの主な仕事は文章の入力，書式の設定，出力であって，文章の推敲を積極的に助けるわけではない．

ワードプロセッサでつくられた文書は機械可読な形で蓄えられている．それにもかかわらず清書出力にしか利用されないというのではつまらない．計算機でその中に何とかふみ入って推敲に役立つ情報を取り出してみたいという動機で開発を始めた．

２．２　プロトタイピング

推敲に役立つ情報とは何か，計算機によってどんな情報をとりだせるのか，計算機を用いることによって従来と違った推敲作業が可能になるのか，など必ずしも明らかにされているわけではない．すなわち推敲を支援するソフトウェアを作ろうというのにその仕様が明確になっていない．とくに，推敲作業のためにはどのようなユーザインタフェースが相応しいのか分からない．

日本語文章の推敲支援ツールの仕様を考える上で英語文章の推敲を助けるＩＢＭのＥＰＩＳＴＬＥやＵＮＩＸのＷｒｉｔｅｒ’ｓ　Ｗｏｒｋｂｅｎｃｈなどが参考になる．しかし，日本語には，分かち書きされていない，字種が多いなどの日本語特有の問題点が多く，英語文章に対するものをそのまま使うわけにはいかない．そこで「推敲」の開発に当たってプロトタイピングの手法を採用することにした．

３．プロトタイプ０（ＰＴ０）

３．１　開発方針

プロトタイプ０（ＰＴ０と略記）は，準備段階である．推敲に役立つと思われる機能をとりあえず実現して試してみることにした．試作に際して以下の方針で臨んだ．

要求：　推敲支援ツールは文章中に問題になりそうな箇所があればそれを指摘することができればよい．推敲するのは書き手であって計算機ではない．実用を考えれば，我々が普段書く程度の長さの文章（例えば学会誌刷上がり８ページ程度）を待ち遠しくない時間内で処理してほしい．

制約：　推敲作業を支援するには，単語処理，文法処理を行うことが有効な方法と考えられる．しかし，先に述べた日本語文章特有の問題点のため，まず形態素解析のところで苦労した挙句に，解析結果にあいまいさが生ずる可能性があること，実用規模の文章の構文解析を行おうとすると解析時間がかかりすぎることが予想される［６］．そこで，辞書を使わず文法的な解析も行わずに入力文章を字面だけで解析する方法を採る．

方法：　日本語文章は分かち書きされていないので辞書なしに単語を識別することは困難である．そこで「推敲」では句点（。）に着目して文を切り出す，字種の変わり目により文字列を切り出す，という２つの解析方法を基本とする．

外部仕様：　推敲支援ツールは，九州大学大型計算機センターＦＡＣＯＭ　Ｍ３８２　ＯＳ　Ⅳ／Ｆ４ＪＥＦのＦＤＭＳ和文エディタで作成された文章を解析対象とする．解析結果はとりあえず，九大センターに設置されている日本語ラインプリンタに出力することにする．（以降，ＦＤＭＳ和文エディタで作成したファイルをＦＤＭＳファイルと呼ぶ．）

言語：　ＪＥＦのもとではＣＯＢＯＬ，ＦＯＲＴＲＡＮ，ＰＬ／Ｉで日本語文字を扱うことができる．しかし，ＦＤＭＳファイルには，日本語文字（２バイト）と従来の英数文字（１バイト）がシフトコードを境界として自由に混在しているため，ＦＤＭＳファイルの処理手続きを上記のプログラミング言語で記述することはかなり面倒である．これではプロトタイピングには向かない．

一方，ＳＮＯＢＯＬ４は挿入・削除を伴った文字列照合や逆戻りを含む文字列に関する複雑な操作を簡単に記述できる．従ってテキストファイル処理などに際し適当なツールがない場合，その場でプログラムをつくってみるのに適している［７］．我々の研究室ではＦＤＭＳファイルを扱えるような機能拡張をこの言語に行った．これを日本語ＳＮＯＢＯＬ４［８，９］と呼ぶ．この言語を推敲支援ツールのプロトタイピングに用いることにした．

３．２　実現

ＰＴＯでは，以下の４種のプログラムを作成した．

SENTENCE:　句点などに着目して文を切り出す．
KWIC:　字種別（漢字，カタカナ，ひらがな，英字）に文字列のKWICを作成する．
XREF:　字種別に文字列とレコード番号との相互参照表を作成する．
PAREN:　括弧の対応を調べる．

SENTENCEプログラムの出力例を図１に示す．

このプログラムは文末の単調性や長すぎる文の検出に役立つ．なお，図１の出力例は見やすくするため，プロトタイプⅢの版の出力例を使っている．

４．プロトタイプＩ（ＰＴＩ）

４．１　ＰＴＯの評価

ＰＴＯを使ってみると，KWICやXREFプログラムの出力結果が膨大となり結果を調べるのが大変である

```
NO. レコード　　文頭　　文末　　文長
 1 01004000 日本語文章推敲支援ツール---ツールのプロトタイピング.:8
 2 01005000 太島和州はフ尖大じ卜語卜か尹志---並順二.尹志州大学工学部).:+
 3 01006000 牛（九日工学部）　　--　　（九州大学）.:+
 4 01007000 1．ソ日プちれ推の本　　--　　1さ，きこそにこと開紹と開発介に「開発支援ツール」は，[が発介に「開発支援ツール」は，[多しすす推敲支援ツール]].:+
 5 01008000 　フ本ロら敲よ報」う告推敲開発機　　--　　2．開てはてら始めピング.:+
 6 01009000 　州はじ語主ワプそプ役推敲宝本敵発動機　　--　　2．開てはてら始めピング.:+
 7 01010000 　和大語主ワプそプ役推敲宝本敵発動機　　--　　2．開てはてら始めピング.:+
 8 01011000 　和大語主ワプそプ役推敲宝本敵発動機　　--　　2．開てはてら始めピング.:+
 9 01012000 　和大語主ワプそプ役推敲宝本敵発動機　　--　　2．開てはてら始めピング.:+
10 01013000 　和大語主ワプそプ役推敲宝本敵発動機　　--　　2．開てはてら始めピング.:+
11 01014000 　和大語主ワプそプ役推敲宝本敵発動機　　--　　2．開てはてら始めピング.:+
12 01015000 　和大語主ワプそプ役推敲宝本敵発動機　　--　　2．開てはてら始めピング.:+
13 01016000 　和大語主ワプそプ役推敲宝本敵発動機　　--　　2．開てはてら始めピング.:+
14 01017000 　和大語主ワプそプ役推敲宝本敵発動機　　--　　2．開てはてら始めピング.:+
15 01018000 　和大語主ワプそプ役推敲宝本敵発動機　　--　　2．開てはてら始めピング.:+
16 01019000 　和大語主ワプそプ役推敲宝本敵発動機　　--　　2．開てはてら始めピング.:+
17 01020000 　和大語主ワプそプ役推敲宝本敵発動機　　--　　2．開てはてら始めピング.:+
18 01021000 　和大語主ワプそプ役推敲宝本敵発動機　　--　　2．開てはてら始めピング.:+
19 01022000 　和大語主ワプそプ役推敲宝本敵発動機　　--　　2．開てはてら始めピング.:+
20 01023000 　和大語主ワプそプ役推敲宝本敵発動機　　--　　2．開てはてら始めピング.:+
21 01024000 　和大語主ワプそプ役推敲宝本敵発動機　　--　　2．開てはてら始めピング.:+
22 01025000 　和大語主ワプそプ役推敲宝本敵発動機　　--　　2．開てはてら始めピング.:+
23 01026000 　和大語主ワプそプ役推敲宝本敵発動機　　--　　2．開てはてら始めピング.:+
24 01027000 　和大語主ワプそプ役推敲宝本敵発動機　　--　　2．開てはてら始めピング.:+
25 01028000 　和大語主ワプそプ役推敲宝本敵発動機　　--　　2．開てはてら始めピング.:+
26 01029000 　和大語主ワプそプ役推敲宝本敵発動機　　--　　2．開てはてら始めピング.:+
27 01030000 　和大語主ワプそプ役推敲宝本敵発動機　　--　　2．開てはてら始めピング.:+
28 01031000 　和大語主ワプそプ役推敲宝本敵発動機　　--　　2．開てはてら始めピング.:+
```

図１．SENTENCEプログラムの出力例

こと，出力結果を取りに行く際に作業が中断されてしまい使い勝手が悪いこと，出力結果が分かりにくいこと等の問題点が見つかった．

しかし，そのような問題点はあったものの我々の研究室で実用の目的に使用して，文末の単調性や表記のゆれなどを検出するのに役立つことがわかった．

４．２　機能の追加と確定

ＰＴＩでは機能の追加と確定とを図った．まず，ＰＴＩでは，結果をディスプレイに表示するように変更し，結果の検索のための機能を追加することにした．その際，出力結果の形式を変更し，出力が見易いように配慮した．さらに，ＰＴＯの４種のプログラム以外に，推敲作業に必要と考えられる機能を実現の容易なものから増やしていった．表１にＰＴＩの処理の内容を簡単にまとめておく．具体的な機能の内容については文献［２］を参照されたい．

表１．ＰＴＩの機能概要

----------	日本語保存ファイルを処理するコマンド
SENTENCE	文を抽出し文頭，文末，文長を並べて表示する．
PASSIVE	受動態を捜す．
KOSOA	指示詞に下線を引いて日本語ＬＰにフォーマット出力する．
STRING	各種文字列を抽出しＳＴＲＩＮＧファイルを作成する．
STRUCT	段落の最初の文と最後の文を残し他の文は各文字を「－」で置き換え日本語ＬＰにフォーマット出力する．
HEADING	見出しとなる文を抽出する．
PAREN	括弧の数を数える．
RENUMBER	１文／１レコードに直して（ＦＤＭＳファイルに）出力する．
----------	ＳＥＮＴＥＮＣＥの結果ファイルを処理するコマンド
R-SENTENCE	文末からソートする．
----------	ＳＴＲＩＮＧの結果ファイルを処理するコマンド
KWIC	ＫＷＩＣファイルを作成する．
R-KWIC	逆ＫＷＩＣファイル（キーワードを末尾順からソート）を作成する．
COUNT	キーワードの長さ，数，割合の一覧表を作る．
LENGTH	指定した範囲の長さの文字列を表示する．
S-INDEX	キーワード索引を作成する．
XREF	キーワードとレコード番号のクロスリファレンスを作る．
----------	ＫＷＩＣの結果ファイルを処理するコマンド
K-COUNT	キーワードの長さ，数，種類，割合の一覧表を作成する．
INDEX	キーワード索引を作成する．
MERGE	２つのＫＷＩＣファイルをマージして１つにする．
----------	ＩＮＤＥＸの結果ファイルを処理するコマンド
FREQUENCY	キーワード索引を頻度順にソートする．
----------	その他のコマンド
EXIT	コマンドを終了する．
NPR	結果の出力先（プリンタ名）を指定する．
TSS	ＴＳＳコマンド実行モードに移る．
LIST	結果を出力する．
RENAME	結果のファイルを別の名前に変更する．
COPY	結果のファイルを別のファイルにコピーする．
DEL	結果のファイルを削除する．

4．3　中間ファイル

機能を増やしていく段階で，KWICやXREFにかなり時間がかかっていたので原因を調べたところ入力ファイルから文字種毎に文字列を切り出す部分にかなり実行が集中していることがわかった．KWICやXREFを行う毎に文字列を切り出すのはムダである．文字列を切り出す部分だけを独立させ切り出した文字列を中間ファイルとして保存し，その中間ファイルに対して処理を行うようにしたらよい．そこで，中間ファイルを作成するためのプログラムSTRINGを追加するとともに，その中間ファイル（STRINGファイルと呼ぶ）を処理するようにKWICやXREFを変更した．さらに，このSTRINGファイルやKWICの結果ファイルを処理するプログラムを新たに追加した．

図２に各プログラムと入力ファイル，出力ファイルとの関係図を示す．各プログラム名の上の欄のファイルを入力ファイルとする．例えば，STRINGプログラムはＦＤＭＳファイルを入力し，STRINGファイルを出力する．また，KWICプログラムはSTRINGファイルを入力としKWICファイルを出力する．

4．4　コマンドプロシジャ

表１のプログラムをＴＳＳ環境で実際に動かすには，一連のＴＳＳコマンドを次々に発行してファイルの割当や解放を動的に行う必要があるので試作段階では非常に煩わしい．

日本語ＳＮＯＢＯＬ４ではプログラム中からＴＳＳのコマンドを発行できるような機能を備えているため，ＰＴ０ではファイルの割り当て等をすべてプログラム中で行い，実行の際には日本語ＳＮＯＢＯＬ４の処理系を起動するだけにしていた．ＰＴＩでは，ファイルの割り当て等の部分を分離してコマンドプロシジャとし，日本語ＳＮＯＢＯＬ４のプログラムは文字列の処理だけを行うようにした．また，プログラムとコマンドを一対一に対応するように配慮した．従って，もしプログラムの変更などがあれば，そのプログラムだけを書き変えればよいことになった．

5．プロトタイプⅡ（ＰＴⅡ）

5．1　ＰＴＩの評価

ＰＴＩの開発により推敲支援ツールに必要な機能をほぼ確定することができた．また，それらの機能を実現するためのプログラムとファイルの構成，即ち，ソフトウェア構成が明らかになった．

しかし，前述したように，日本語ＳＮＯＢＯＬ４は文字列処理を得意とするため，「推敲」の試作や機能確定に適するが，解釈実行型なので処理速度が遅い．表２にいくつかのコマンドの処理時間（ＣＰＵ時間）を示す．ＦＡＣＯＭ　Ｍ３８２を利用しているかぎり例に用いた程度の大きさの文章について大抵のコマンドの場合，待ち遠しくない時間内で処理できるといえる．しかし，文章Ｃ，Ｄに対するコマンドSTRINGやKWICの実行にはかなり時間がかかっており，満足できるとはいい難い．

5．2　処理速度の改善

ＰＴⅡでは，ＰＴＩで確定した機能の処理速度を改善し，実用性を高めることを目指した．日本語ＳＮＯＢＯＬ４で書かれたプログラムをＦＯＲＴＲＡＮ７７で書き換えることにより性能改善を図った．

日本語ＳＮＯＢＯＬ４では，挿入，削除を伴った文字列照合を一文で書け，それらの文の組み合わせにより文字列処理が容易にできる．しかし，ＦＯＲＴＲＡＮ７７は日本語ＳＮＯＢＯＬ４のような文字列処理機能をほとんど持っていない．ＦＯＲＴＲＡＮ７７で書き替える際に，日本語ＳＮＯＢＯＬ４の各々の文字列照合機能を関数化し，表現するのも一つの方法であるが，ＰＴⅡでは，一層，高速化を図るため元のプログラムのアルゴリズムには必ずしもとらわれないことにした．

具体的にはプログラムの性質に応じて以下の３種類の手法を用いた．

KOSOA,SENTENCE：　これらのプログラムの処理時間はほとんど日本語文章テキスト上の文字列照合に費やされている．しかし，日本語文章テキストについ

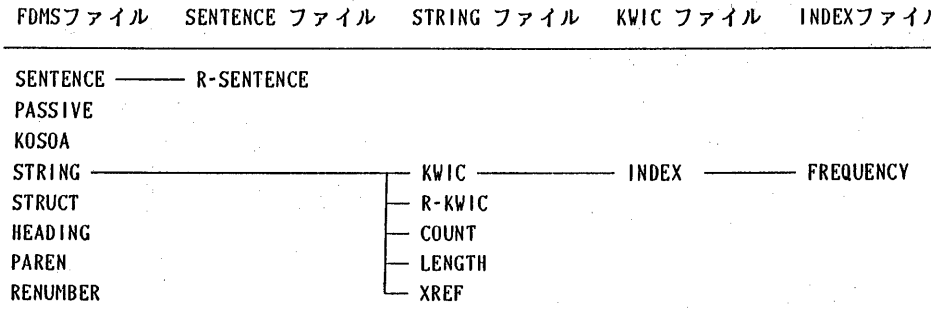

図２．ＰＴＩにおけるプログラムとファイルの関係図

表２．各コマンドの処理時間（単位ms）

文章（文字数）	SENTENCE	KOSOA	R-SENTENCE	STRING	KWIC 漢字	KWIC ひらがな	COUNT ひらがな	INDEX 漢字
A（5007）	1039	395	272	4370	5845	5610	368	1722
B（9842）	1866	654	381	9730	9790	10752	725	2726
C（23583）	5991	1535	1455	38707	39425	64496	2179	11701
D（36605）	17741	2598	3526	75519	46484	94253	3849	11735

表３．改善されたコマンドの処理時間（単位ms）

文章（文字数）	SENTENCE	KOSOA	STRING	KWIC 漢字	KWIC ひらがな	INDEX 漢字
A（5007）	58	107	299	193	205	55
B（9842）	74	122	455	317	354	66
C（23583）	167	196	953	793	943	118
D（36605）	272	278	1301	887	1042	128

ては，それ特有の性質に着目した効率のよい文字列照合アルゴリズムが開発されているわけではない．そこでとりあえず，英文テキスト上でその有効性が確認されている高速文字列照合アルゴリズム［１０］を採用することにした．

STRING：　文字コード系の特性を用いて字種の分類を行なった．

KWIC,INDEX：　日本語ＳＮＯＢＯＬ４プログラムのアルゴリズムをそのまま利用した．

書き換え後のコマンドの実行時間の例を表３に示す．表２と比較して，例えば，KOSOA コマンドでは，約１０分の１の処理速度の改善が得られた．また，最も時間がかかっていたSTRINGコマンドの場合，文章Ｄについて５０分の１以下の時間短縮が実現できた．

６．プロトタイプⅢ（ＰＴⅢ）

６．１　ＰＴⅡの評価

ＰＴⅡにより各コマンドの処理時間が大幅に短縮され，「推敲」は実用システムのレベルになった．

しかし，逆に処理速度が向上しただけ，ユーザインタフェースに目が向くようになり，以下のようなＰＴＩでの問題点が再浮上してきた．

ＰＴＩの問題点：　STRINGコマンドの結果でき上がるSTRINGファイルは，以後の処理を容易にするための中間ファイルである．ＰＴＩの段階では，STRINGファイルの作成には最も時間がかかっていたので（表２参照），処理時間の節約を考え，一度STRINGファイルを作っておけば何度でも使えるようにしていた．しかし，STRINGファイルは字種別に存在するので，利用者は複数個のSTRINGファイルを管理する手間が増える．しかも，ＰＴＩではSENTENCEやKWICなどの結果ファイルを処理するコマンドが追加されたので（図２参照），利用者はSTRINGファイルだけでなく，それらのファイルも管理しなければならない．

６．２　コマンドの再構成

ＰＴⅡの開発により，STRINGコマンドが短時間で実行可能になった結果，中間ファイルを利用者に見せる意義が減少した．ＰＴⅢでは，STRINGファイルを隠して，利用者が目的コマンドを直接実行できるように，ファイル，コマンドの再構成を行うとともに，ユーザインタフェースの改良を行なった．その際，STRINGファイルを含めた結果ファイルを「推敲」システム側で管理することにしたので，利用者は日本語文章の格納されているソーステキストファイル（ＦＤＭＳファイル）だけを管理すれば良い．ＰＴⅢのコマンド一覧を表４に示す．表４のコマンドは全て単独で用いることができ，ＰＴＩのような発行順序の制限がない．

表４．ＰＴⅢのコマンド体系

日本語テキストファイルを処理するコマンド			その他
STATISTIC	KWIC		DSNAME
SENTENCE	R-KWIC		MACRO
R-SENTENCE	COUNT		MLIST
PASSIVE	LENGTH		EXIT
KOSOA	XREF		NPR
STRUCT	INDEX		NEDIT
HEADING	FREQUENCY		TSS
PAREN			LIST
RENUMBER			

ＰＴⅡとＰＴⅢの主な変更点を以下にまとめる．

廃止，統合，追加

STRINGファイル及び結果ファイルを「推敲」側で管理することにより以下のコマンドを廃止，統合，追加した．

・廃止：　STRING,MERGE,RENAME,COPY,DELETE
・統合：　COUNT とK-COUNT,INDEX とS-INDEX
・追加：　DSNAME

機能強化

次のコマンドを追加し「推敲」の機能を強化した．
STATISTIC,MACRO,MLIST,NEDIT

出力形式の改良

R-SENTENCE,PAREN,KOSOAコマンドの出力形式を見易いように変更した．この他，コマンドの入力をメニュ方式にするなどして全体的にユーザインタフェースを改善した．

精度の向上

PASSIVE コマンドの処理方式を改良して受け身形の検出精度を向上させた．

利用者の意見の収集

MESSAGE コマンドを設けて利用者の意見が「推敲」作成者に伝わるようにした．

７．まとめ

現在も「推敲」の改善作業中であるが，プロトタイプⅢまでの作業の経過及びそれから得られた知見を以下にまとめる．

（１）プロトタイプ０からプロトタイプⅢまでの経過を図３に示す．通常のプロトタイプでは，図３中のプロトタイプＩでプロトタイプの段階を終了し，プロトタイプⅡに相当する段階では実用システムを作成するという経過をたどる．我々も当初そのように計画し，ＰＴⅡではＳＮＯＢＯＬ４プログラムをＦＯＲＴＲＡＮ７７で書き換え，実用レベルの「推敲」システムを作成した．しかし，６．２節に述べたように処理速度の改善がユーザインタフェースの変更をもたらす結果となった．これは従来のプロトタイピングではあまり指摘されていない点である．

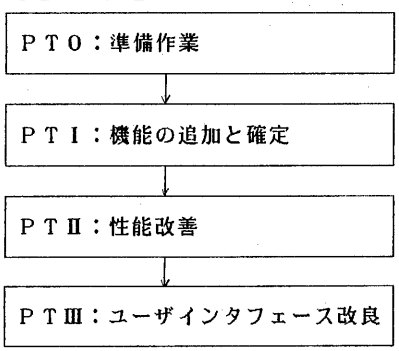

図３．プロトタイピングの経過

図３のような経過をたどった理由の一つはＰＴ０の段階からプロトタイプを作る際に性能は別にして，その範囲で実用に供しうるように作っていったことによると思われる．プロトタイプと性能は相入れない．しかし，ユーザの側から見ればプロトタイプの反応の悪さがユーザインタフェースの評価につながったとしても不思議はない．プロトタイピングにはユーザの協力が必要である．今後，プロトタイプとその性能との関係を考慮したプロトタイピングの手法についても検討する必要があろう．

（２）「推敲」を構成するＳＮＯＢＯＬ４プログラムは，早いものでは２～３時間，複雑なものでも３～４日ですべて試作することができた．

このように比較的簡単に作成できたのは，ＳＮＯＢＯＬ４の強力な文字列操作能力により文字列の照合アルゴリズムを全く気にせずプログラミングできたことがその大きな理由である．文字列操作能力以外に，このツールのプロトタイピングに寄与した日本語ＳＮＯＢＯＬ４の機能の主な点を以下にまとめる．

（２ａ）ＦＤＭＳファイルの入出力・・・このファイルの仕様は明らかにされておらず，他言語でこのファイルを扱うのは大変であるけれども，日本語ＳＮＯＢＯＬ４では一般のファイルと全く同様に区別せず取り扱うことができるようにしている．

（２ｂ）日本語テキスト処理に便利なキーワードの追加・・・日本語テキストを処理する際に文字種の情報は役にたつ．日本語ＳＮＯＢＯＬ４では，第一水準漢字，第二水準漢字，ひらがな，カタカナ等の文字集合を表わすキーワードを用意している．これにより文字列の切り出しを容易に行うことができる．

（２ｃ）ＴＳＳ環境での使い勝手・・・４．４節でも述べたように日本語ＳＮＯＢＯＬ４からＴＳＳコマンドを発行できるようになっていたので特にプロトタイピングの初期において役にたった．

（２ｄ）実行回数計数機能・・・日本語テキスト処理機能を追加する以前に我々の研究室で，実行回数計数機能をＳＮＯＢＯＬ４処理系に加えていた．［１１］この機能は，プログラムを作成する際（特にプログラムのデバッグ）に役に立った．

（３）ＰＴⅡにおけるＦＯＲＴＲＡＮへの書き換え作業は文字列照合アルゴリズムの習得期間を除けば非常に短期間で行なうことができた．これは次の二

つの理由によるものと考えられる．

（３ａ）４．４節で述べたように，「推敲」システムを構成する各々のコマンド（プログラム）を相互独立性を持つように設計していたため，プログラムの部分的な置き替えが可能であった．すなわち，システムのユーザインタフェースやアルゴリズムを全く変更せずに処理時間の遅いいくつかのプログラムを書き替えるだけの作業で高速化を図ることができた．一方，中にはアルゴリズムを変更したプログラムもあったが，ＰＴＩのプロトタイピングによって機能が定まったものに関する作業であったため比較的簡単な作業であった．

（３ｂ）日本語ＳＮＯＢＯＬ４処理系の実現の際に入出力を行うための専用のＩ／Ｏプリミティブが設計された［１２］．これは，日本語ＳＮＯＢＯＬ４以外のプログラムからも容易に呼び出しができるようになっている．Ｉ／Ｏプリミティブを利用できたことが書き換え作業を著しく容易にした．

８．おわりに

日本語ＳＮＯＢＯＬ４で書いたプログラムでは長時間を要したコマンドが書き換えにより高速化されたことによって，「推敲」はより長い文章を処理対象とすることが可能になった．この場合，長い文章から抽出した推敲情報は非常に膨大になるので，どんな方法で推敲情報を利用者に適確に選んでもらうかという問題も新たに検討すべき課題となる．また，STRINGコマンドの高速化と共にユーザインタフェースの改善方法として中間ファイルを利用者から隠した（ＰＴⅢ）けれども，対象の文章がかなり長い場合は中間ファイルを見せる必要性が再度出てくるかもしれない．今後，ファイルやコマンド構成の変更，改善がないとはいえない．

３．１節で述べたように，「推敲」を試作するに当たって，自然言語処理研究の成果をほとんど利用していない．推敲支援というからには，係り受けの多義性や，並列句の指摘をしてくれると役に立つにちがいない．また辞書［１３］の使用も試みつつある．しかし，表示情報の爆発や処理時間の増大を懸念していずれも本格的に利用しようという段階にいたっていない．

現在このツールは暫定版ではあるが，九大大型計算機センターで一般に公開しており［２］，少し広い範囲で使ってもらい，利用者からの意見や要望を取り入れながらさらに改良したいと考えている．また，他の計算機システムへの移し換えも進行中である．

参考文献

［１］Special Issue On Rapid Prototyping, ACM SIGSOFT SOFTWARE ENGINEERING NOTES, Vol.7, No. 5, 1982

［２］牛島，日並，尹，日本語文章推敲支援ツール「推敲」の使用について，九州大学大型計算機センター広報，Ｖｏｌ．１８，Ｎｏ．１，１９８５

［３］牛島，日並，日本語文章推敲支援ツールの試作とその作成環境，第３５回ソフトウェア工学研究会（３５－２）１９８４

［４］日並，牛島，日本語文章推敲支援ツールのプロトタイピング（１）－日本語ＳＮＯＢＯＬ４の利用，情報処理学会第２９回全国大会論文集（３Ｄ－８），１９８４

［５］尹，高木，牛島，日本語文章推敲支援ツールのプロトタイピング（２）－性能の改善について，情報処理学会第２９回全国大会論文集（３Ｄ－９），１９８４

［６］吉村，日高，吉田，文節数最小法を用いたべた書き日本語文の形態素解析，情報処理学会論文誌，Ｖｏｌ．２４，Ｎｏ．１，ｐｐ．４０－４６，１９８３

［７］角田，ＳＮＯＢＯＬ，情報処理，Ｖｏｌ．２２，Ｎｏ．６，ｐｐ．４７３－４７６，１９８１

［８］吉田，牛島，日本語ＳＮＯＢＯＬ４の使用について，九州大学大型計算機センター広報，Ｖｏｌ．１７，Ｎｏ．３，ｐｐ．１１０－１３４，１９８４

［９］Ushijima, K., et al., SNOBOL4 with Japanese text processing facility, Proc. ICTP' 83, pp.235-240, 1983

［１０］Aho, A.V. and Corasick, M.J., Efficient String Matching : An Aid to Bibliographic Search, Comm. ACM, Vol.18, No.6, pp.333-340, 1975

［１１］吉田，牛島，実行回数計数機能を追加したＳＮＯＢＯＬ４処理系とその移し換えについて，情報処理学会論文誌，Ｖｏｌ．２４，Ｎｏ．６，ｐｐ．７８９－７９５，１９８３

［１２］吉田，牛島，日本語ＳＮＯＢＯＬ４処理系のためのＩＯプリミティブの設計と実現，情報処理学会第２８回全国大会論文集（２Ｈ－７），ｐｐ．３５５－３５６，１９８４

［１３］吉田，日高，稲永，田中，吉村，公用データベース日本語単語辞書の使用について，九州大学大型計算機センター広報，Ｖｏｌ．１６，Ｎｏ．４，ｐｐ．３３５－３６１，１９８３