ソフトウェア　工学　12－2
（1979. 10. 26）

簡易言語プログレスによる生産性向上について

魚田勝臣　小碇暉雄　溝口徹夫　富沢研三
三菱電機株式会社

1．はじめに

オフィス・コンピュータは、昭和52年度26%，53年度32%の非常な勢いで伸びており、第三次普及期を迎えている。
この背景には、LSIに代表される半導体技術の進歩やディスクの大容量化などハードウェアの技術革新とともに、高性能化、低価格化が促進され、小さな企業でも使えるようになってきたことがあげられる。

しかし、この旺盛な需要に支えられた高度成長を阻害する要因として、ユーザ・ソフトウェアを開発するシステム・エンジニアの不足が顕著になってきた。
この問題を解決するには、生産性の高い言語が必要である。しかし、既存の言語は、汎用機用に開発されたものをオフィス・コンピュータ用に改造したものが多く、最近のオフィス・コンピュータにおける技術的進歩に追随していないため、オフィス・コンピュータを指向した、生産性の高い言語を開発する必要があった。

このような背景のもとに、我々は、従来から研究してきた簡易言語を集大成した「プログレス」を開発し、多くのユーザに提供して生産性の高さを実証することができた。この論文では、オフィス・コンピュータMELCOM80用に開発した簡易言語プログレスについて、言語の設計思想，言語の概要，及び生産性に重点を置いて報告する。

2．簡易言語と生産性

データ処理用の簡易言語は、ソフトウェアの生産性を上げるために開発されたものである。
簡易言語を大別すると
①手続き向き言語の流れを組むもの。
例：COBOLの書き方を簡略化する試みとしての簡易COBOL
②非手続き向き言語の流れを組むもの。
例：RPGⅡのようなFill in the blank 形式の言語
に分類できる。　②については、特にオフィス・コンピュータでの発展が著しい。

又、別な面からのとらえ方として、事務データ処理をパターン化し、それぞれの業務や機能をパッケージとして開発する試みがある。
このパッケージを大別すると
①ある業種のシステム全体を包含した業種分離型パッケージ
②業種間で共通の業務をまとめた業務分離型パッケージ
③それぞれの業種で共通に行われる機能を集約させた機能分離型言語
に分類できる。

図1は、これらの簡易言語やパッケージの関連をまとめたもので、前に述べた手続き向き言語や非手続き向き言語を、機能包括型言語と表現してある。

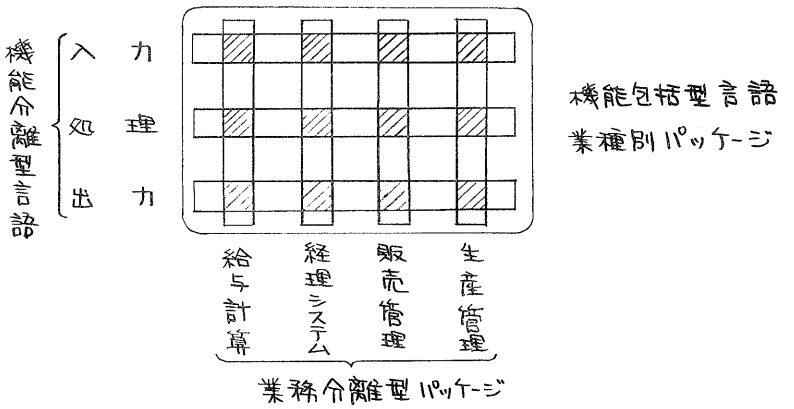

図1. 簡易言語の関連

これらの言語やパッケージの生産性については、

①業種別パッケージ
②業務別パッケージ
③機能分離型言語
④機能包括型言語

の順になることは言うまでもない。しかし汎用性については、これと全く逆の順になり、特にパッケージは非常に多くの種類が必要となってくる。

3. プログレスの開発方針

プログレスの開発の背景にあるのは、ユーザ・ソフトウェアを開発するためのシステム・エンジニアやプログラマ不足の緩和であり、最終目標は全くの素人でもシステムが組めるようにすることである。

プログレスの開発にあたり、我々は次の方針を決めた。

①コンピュータの素人にも簡単に覚えられる言語であること。
②言語の習得からシステム設計、コーディング、デバッグ、保守に至る総合的な生産性の向上が実現できること。
③オフィス・コンピュータ特有の対話処理を得意とする言語であること。
④適用分野が広いこと。
⑤ハードウェア環境は、近い将来も含めてオフィス・コンピュータに限定すること。

4．生産性向上のための方策

3章で述べた目標を実現するために、ソフトウェア開発の手順と各々の過程における生産性向上のための着眼点及び対策について考慮し、これを表1に示す。

表1．ソフトウェア開発における生産性向上のための方策

NO.	ソフトウェア開発の手順	プログレスで行った生産性向上のための方策
1.	言語を覚える。	1.1 英語的な表現を避ける。 1.2 覚え易い、意味のある記号を用いる。 1.3 バッチ処理と、インライン，オンライン処理とでは、内部処理ロジックを分ける。
2.	プログラム仕様の検討。	2.1 記入シートを8枚に集約する。 2.2 ファイルや画面のレイアウトを決めるだけで、特に流れ図を作成しなくてもコーディングできる。
3.	コーディング	3.1 記入項目を極力減らすことにより、生産性の向上とミスの可能性を減らす。 3.2 記入シートに工夫をこらし、マニュアルなしでコーディングできるようにする。 3.3 内蔵機能を多くする。 3.4 よく用いるロジックはマクロ命令化する。
4.	リース・プログラムの作成	4.1 対話形式でどの端末からでも簡単にリース・プログラムを作成できる。 4.2 同じファイルや画面を扱う場合、それらを流用するためのCOPY機能を持つ。
5.	コンパイル	5.1 コンパイル・スピードを上げる。 5.2 コンパイル時にリース・プログラムの修正が行なえる。 5.3 リース・プログラム作成後、どの端末からでも即座にコンパイルできる。
6.	デバッグ	6.1 エラー・メッセージをカナ文字で表現する。 6.2 対話形式でリース・プログラムの修正が行なえる。 6.3 デバッグ用のテンプレートを作る。 6.4 1台の端末だけで複数の端末を扱うプログラムのデバッグができる。
7.	保　守	7.1 記入用紙をそのままドキュメント資料として使える。 7.2 プログラムの仕様変更が容易である。 7.3 入力項目の定義とそれに対する処理を同時に記入できるので、誰が見ても理解し易い。

プログレスでは表1に示すソフトウェア開発の全ての手順に対して、生産性向上を総合的に検討しそれを言語仕様に反映させている。

5. プログレスの概要

我々は高い生産性を有する言語として、2つのプログレスを開発した。1つは、バッチ処理を主体とした機能分離型の言語であり、もう一つは、それを包含し更にインライン，オンライン処理を得意とする機能包括型の言語（プログレスII）である。いずれもFill in the blank形式の言語で、ここでは特にプログレスIIを中心に述べる。

5.1 プログレス（機能分離型）

プログレスは図1に示すように処理形態を分離し、入力システムと出力システムといったように言語を構築したものである。処理形態としては、LOAD&GOの形をとっている。処理形態を細分化したために、記入シート（指示書と呼んでいる）の種類が増えている。

しかし、このシステムは、簡易言語のコンクールでも優秀な成績を収め、昭和50年に発表して以来、多くの顧客で使われている。

5.2 プログレスII （機能包括型）

プログレスIIは次に示す8種類の指示書で構成されたどんな処理にも対応できる汎用簡易言語である。

(1) プログラム指示書 ---- プログラムの概要について記入する。
(2) 端末指示書 ------- オンライン処理で用いる端末について記入する。
(3) 入力ファイル指示書 --- 入力及び更新ファイルについて記入する。
(4) 出力ファイル指示書 --- 出力及び拡張ファイルについて記入する。
(5) プリント指示書 ------ 印刷装置に対する出力形式について記入する。
(6) 端末入出力指示書 --- 端末との入出力データについて記入する。
(7) 作業項目指示書 ----- プログラムで用いる定数やテーブルを記入する。
(8) 処理指示書 -------- 処理内容について記入する。

これらの指示書構成の特徴として次のことがあげられる。

① そのプログラムの処理内容に応じて、必要な指示書を適宜選択すればよい。
② ファイルに関する記述を行う指示書を、その形態に応じて3種類に分け，同じ記入カラムを重複して用いることを避けた。
③ インライン，オンライン処理に対処するため、端末指示書と端末入出力指示書を設けた。
④ 定数やテーブルの定義ができる作業項目指示書を設けた。
⑤ 入力項目を定義する入力ファイル指示書や端末入出力指示書で、入力項目に対するデータの比較やエラー・チェック，合計計算などが行える。
⑥ 処理指示書では、バッチ処理とオンライン処理とでは別々に決められた処理ロジックに沿って記入できる。

次に、プログレスⅡの処理ロジックについてオンライン処理を中心に述べる。

5.2.1 バッチ処理の処理ロジック

バッチ処理の処理ロジックは、RPGⅡのロジックに類似している。つまり、一般的なプログラムの処理形態である 入力 ⇨ 処理 ⇨ 出力 といったサイクルを基本に、レコード識別，マッチング，合計処理，プリント処理などを盛り込んだものである。

5.2.2 オンライン処理の処理ロジック

オンライン処理の処理ロジックは、オフィス・コンピュータで最も多く行われる伝票発行や問い合せ業務をモデルに設計されている。 図2に、このオンライン処理の処理ロジックを示す。

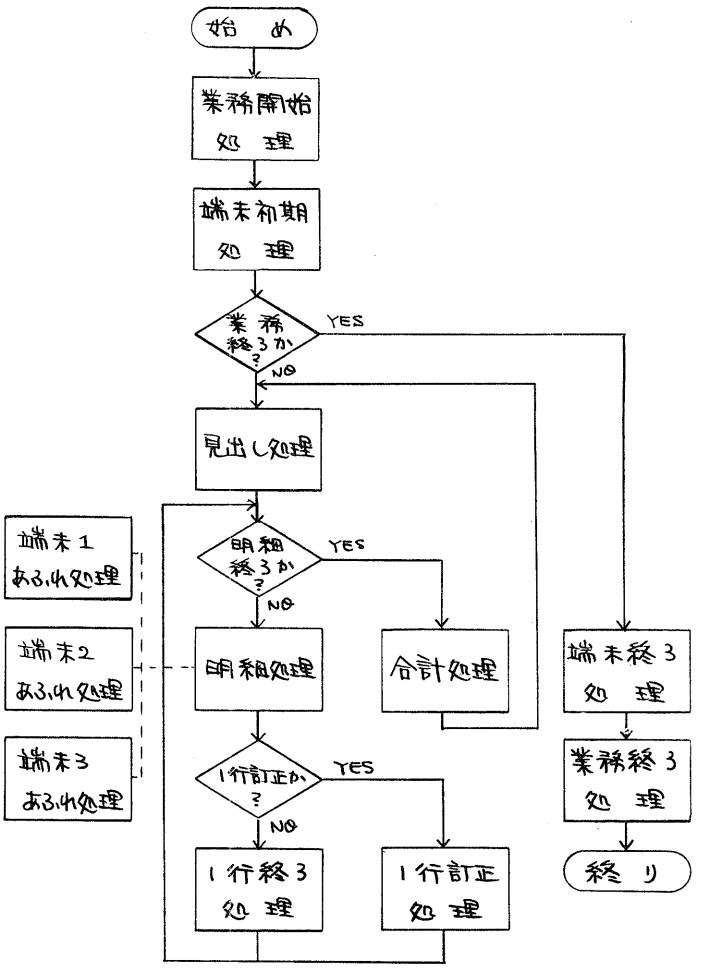

図2．オンライン処理の処理ロジック

次に、プログレスⅡの記入例を紹介する。
これは、図3に示すように、3台の端末からデータを入力しそのデータをディスクに書き込む処理である。　図4はこの処理をプログレスⅡでコーディングした例の一部である。

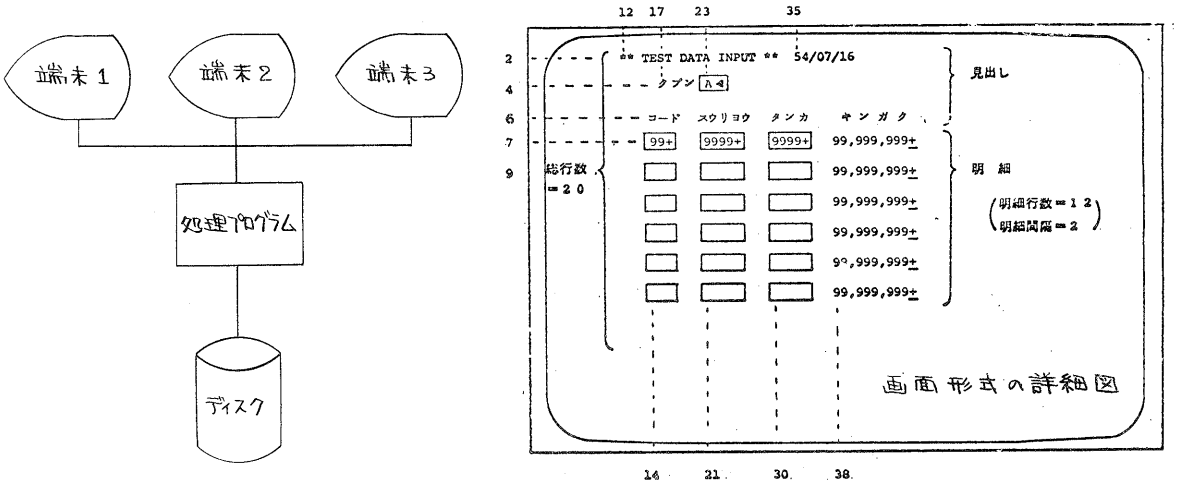

図3．オンライン・プログラムの処理

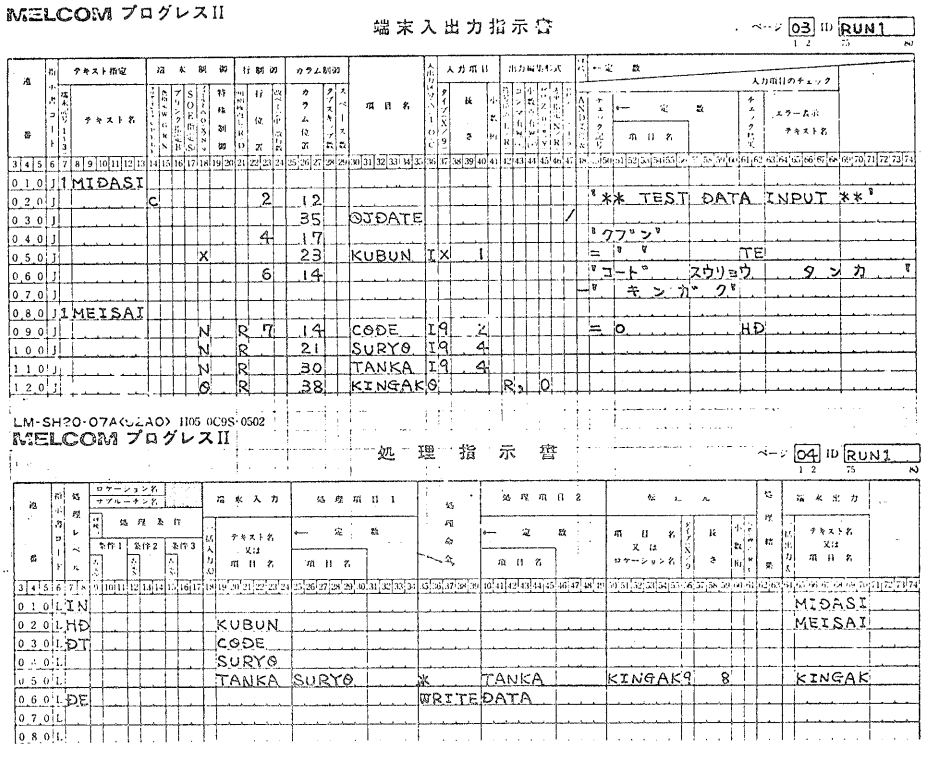

図4．プログレスⅡによるコーディング例

図3に示すような端末を用いる処理をCOBOLやRPGIIでプログラミングする場合、端末に対する専門的な知識を必要とするため、一般にはローカルな端末用の言語(プリ・プロセッサ)を作り、一旦それを通してからコンパイルする方法をとっている。更に複数の端末を1つのプログラムで扱うには、端末の制御などの処理手続きが非常に複雑になってくる。

プログレスIIでは、図4に示すようにシステム設計書(この場合画面のレイアウト)が決ればそれを何行目のどの位置から入出力するか、何回処理を繰り返すか、といった単純な記入でコーディングすることができる。又、端末を多数扱う場合でも端末1台のイメージでコーディングしてよく、複雑なロジックを考える必要がない。

このようにプログレスIIでは、ファイルのレイアウトや画面のレイアウトなどのシステム設計書が決まれば、その数値や記号を記入用紙に記入していくだけでプログラムを作成することができる。これは言語仕様を極力単純化するために、多くの内蔵機能を持っているためで、その代表的なものを表2に示す。

表2. プログレスIIの内蔵機能

項目		内容
バッチ処理	入出力処理	・代表的な処理については、手続きを記入しなくても、自動的に入出力処理を行う。
	レコード処理	・レコード識別やマッチングは、簡単な記号で行なえる。
	合計処理	・項目の自動加算やクリア,合計の出力など自動的に実行する。
	プリント・リスタート	・大量のプリント処理に対してリスタート機能がある。
オンライン処理	端末入出力処理	・項目単位又は複数の項目を一括しての入出力処理を、処理指示書にその名前を記入するだけで、自動的に実行する。
	画面コントロール	・端末入出力指示書で指定された行位置、カラム位置、繰り返しなどの情報をもとに、カーソルの移動を自動的にコントロールする。
	端末制御	・複数の端末の制御を内部で行っている。
	プリント・ファイル	・1台のプリンタを複数の端末よりアクセスする場合、各端末のデータを1ページ単位に集計する。
	1行訂正処理	・伝票発行などで1行訂正できる機能がある。
	エラー・チェック	・端末よりの入力データに対して、エラー・チェック,メッセージの表示、エラー解除を行う。
	レコード・プロテクト	・端末間又はジョブ間におけるレコード・プロテクトを内部で行っている。

6. 生産性の比較

事務処理用言語として代表的なCOBOLとプログレスⅡとで行った生産性についての比較結果を表2に示す。

このテスト結果は、COBOL，プログレスⅡともに未経験者とCOBOL経験9年，プログレスⅡ経験0.5年の経験者に対して行ない、前者はデータを入力してプリント・アウトするバッチ・プログラム，後者は3台の端末を用いてファイル更新を行なうオンライン・プログラムを作成したデータである。

表2．COBOLとプログレスⅡの生産性比較（単位：時間比，ステップ数比）

生産性の項目 ＼ プログラム		未経験者（バッチ・プログラム）		経験者（オンライン・プログラム）	
		COBOL	プログレスⅡ	COBOL	プログレスⅡ
言語の習得		3	1	—	—
プログラム仕様検討		4	1	12	1
コーディング		4	1	6	1
ステップ数	手続き部以外	2.5	1	2.5	1
	手続き部	22.4	1	5.8	1
	総ステップ数	4.3	1	3.6	1
コンパイル回数		7回	5回	11回	4回
コンパイル時間		2	1	2	1
デバッグ時間		3	1	3	1
全体の生産性		4	1	5	1

このデータが示すようにプログレスⅡは言語の習得からデバッグまで全ての面で高い生産性を示している。又、約20本のテスト・プログラムによりCOBOLとの比較を行った結果、COBOL比3－5倍という高結果が得られており、更に、MELCOM 80のユーザにおける使用経験からも同様の報告がなされている。

7. おわりに

本論文では、オフィス・コンピュータMELCOM 80用に開発した簡易言語プログレスの設計思想，言語の概要，生産性について概説した。プログレスは、オフィス・コンピュータの機能をフルに活用でき、極めて生産性の高い汎用簡易言語であり，昭和50年にプログレスを，昭和54年にプログレスⅡを完成して多くの顧客に提供されている。

今後は更に生産性を高める研究を進め，プログレスに反映させていくつもりである。

8. 参考文献

〔1〕 情処学会　第20回全国大会講演論文集　昭54.7.

〔2〕 三菱電機技報　VOL.51 NO.4

〔3〕 オフィス・コンピュータに関する市場調査　昭54.7　(社)日本電子工業振興協会