解　説

事務処理ソフトウェア開発用簡易言語（第4世代言語）の現状と分析†

古　宮　誠　一††

1. はじめに

最近，プログラムの開発現場では，第4世代言語または4GL（=4th Generation Language）と呼ばれる事務処理ソフトウェア開発用簡易言語に期待を寄せる人が多い．実際に使ってみると，使いやすく生産性も向上したというユーザも多く，ソフトウェアを試行錯誤で開発できるという．とすれば，第4世代言語をプロトタイピング支援ツールとみなすこともできるはずである．また，第4世代言語の中には，ソフトウェアの再利用を支援するシステムやプログラム自動生成システムもある．

第4世代言語が対象とするソフトウェアはビジネス・アプリケーションに限られるが，自動プログラミング技術，プロトタイピング技術，ソフトウェア再利用技術，要求定義支援技術などの研究者には，ソフトウェア開発支援ツールとして第4世代言語をもはや無視できない．なぜなら，これらの言語の多くが高い生産性を与えるからである．

第4世代言語は，主としてプログラムの開発現場の要請によって生まれてきたものなので，論文などによる開発者の報告はほとんどない．しかも，そのほとんどが商用化されているが，その実現技術は全く明らかにされていない．これらの言語の中には，これまで見落とされていたソフトウェア開発に必要な要素技術が隠されているかもしれない．

筆者らは，このような理由から第4世代言語の現状を調査した．「第4世代言語の利用の立場」と「第4世代言語の実現方式の立場」から表-1の36種類のパッケージを採り上げ，文献とヒアリングによる調査を実施した．

本稿では，この調査結果を基に，第4世代言語について，①第4世代言語の定義とその機能モデル，②マンマシン・インタフェースからの分析，③第4世代言語の実現方式からの分析，④プロトタイピング支援ツールとしての有効性からの分析，⑤適用範囲や記述能力からの分析，⑥ユーザによる第4世代言語の評価の6点から分析し解説する．また，以上の分析から得られた，第4世代言語の将来方向についても言及する．なお，解説に際しては，発散を避けるために対象を商用化されている第4世代言語のみに絞って議論を進める．

2. 第4世代言語の定義と機能モデル

2.1 第4世代言語の定義とその考察

第4世代言語は下記に示す目的で使われることが多い．

◆オンライン事務処理プログラムの対話型開発支援ツール
◆コーディング量がCOBOLより大幅に少ない言語
◆エンドユーザ言語
◆非手続き型言語
◆オンライン対話型で利用可能な言語
◆データベース操作言語
◆報告書作成用言語
◆プロトタイピング用言語
◆使い方の習得が容易な言語

第4世代言語を定義することは大きな困難をともなう．それは，第4世代言語が上記のように，用途もプログラム構造も異なるソフトウェア・ツールの総称だからである．筆者らは，第4世代言語をその用途から「情報の生成/検索/編集用」の情報生成型 4GL，「プログラム開発用」の業務生成型 4GL，これらの機能を合わせもった統合化 4GL の三つに大別する．

第4世代言語の定義として，次の条件を満たすものとすることがある．

① 非手続き的な記述を支援するプログラム言語で

† The state and Analysis of 4th Generation Language by Seiichi KOMIYA (Information-technology Promotion Agency, Japan).

†† 情報処理振興事業協会技術センター

表-1 第4世代言語の調査結果

項番	4GL 名	開発会社	4GL 種別	システムで使用可能な DBMS	SQL 言語のサポート	4GL の実現方式	プロトタイピング
1	ADS/ONLINE	Cullinet software Inc.	業務生成型	IDMS/R	予定あり	IDMS/R と DD に依存する方式	可能
2	ANSWER/DB	スターリング・ソフトウェア社	情報生成型	IMS, DB 2 他	DB2 で可	DBMS を利用する方式	機能なし
3	AS	IBM	情報生成型	DB 2, SQL/DS	あり	独自の DD (AS-DB) と RDBMS を利用する方式	可能
4	CANO-AID	キヤノン ソフトウェア	業務生成型	ADM, IMS/DB, XDM/SD, DB 2, ADABAS	ADABAS, DB2 で可	部品合成と DD による方法	機能なし
5	CASET	富士通	業務生成型	独自の RDBMS	なし	独自 DBMS の利用と部品合成	可能
6	CSP	IBM	業務生成型	DB 2, SQL/DS, IMS/DB	あり	MSL と ALF とデータフロー部品を利用する方式	可能
7	EASYTRIEVE	Pansophic Systems Inc.	情報生成型	IMS/DB, DL/I, ADABAS, IDMS, TOTAL	なし?	DBMS に依存しない. スケルトン利用方式	可能
8	EASYTRIEVE PLUS	Pansophic Systems Inc.	業務生成型	IMS, AIM, ADM, IDEAL, TOTAL, IDMS/R, ADABAS 他	なし	DBMS には依存しない スケルトン利用の簡易言語	可能
9	ETOILE/OP	日立製作所	情報生成型	RDBF	なし	RDBF に依存する方式, スケルトン利用方式	可能
10	FOCUS	Information Builders Inc.	情報生成型	IDEAL, DB2, IMS, ADM, PDM	あり	独自の DBMS を利用する方式	可能
11	IDEAL/DF	Applied Data Reserch Inc.	業務生成型	IDEAL/DB	あり	IDEAL/DB と DD に依存する方法	可能
12	IDEAL/DQ	Applied Data Reserch Inc.	情報生成型	IDEAL/DB	あり	IDEAL/DB と DD に依存する方法	機能なし
13	IDL II	日本電気	業務生成型	RIQS	検討中	RIQS と DD の利用, スケルトン利用方式	可能
14	JASPOL	日本システム・サイエンス	業務生成型	COBOL, PL/I で扱えるすべての DBMS	なし	DBMS に依存しない, スケルトン利用方式	機能なし
15	JASMAC	日本システム・サイエンス	業務生成型	用意する部品によって決定する	なし	部品定義言語と DD による方法	可能
16	LINC II	米国ユニシス社	業務生成型	DMS II	なし	DMS II と DD に依存する方式	可能
17	MANTIS	Cincom Systems Inc.	業務生成型	SUPRA, TOTAL, DB2, SQL/DS, ADABAS, DL/I	DB2, SQL/DS で可	DBMS を利用する方式	可能
18	MAPPER	米国ユニシス社	情報生成型	MAPPER DBMS	あり	MAPPER DBMS に依存する方法	可能
19	MARK IV	スターリング・ソフトウェア社	業務生成型	IMS, DB2, SQL/DS	DB2 で可	DBMS には依存しない, スケルトン・リポートの生成	機能なし
20	NATURAL	Software AG	統合化4GL	ADABAS	開発中	ADABAS に依存する方式	可能
21	日本語 VAX DATATRIEVE	Digital Equipment Co.	情報生成型	日本語 VAX Rdb, VAX DBMS, RMS	VAXSQL あり	DBMS と DD に依存する方式	可能
22	PLANNER	富士通	情報生成型	独自の RDBMS	なし	独自 DBMS と DD を利用する方式	可能
23	PRO-IV	Pro Computer Science Inc.	業務生成型	IMS/DB, DB2, AIM/DB	あり	VSAM を利用する方式	可能
24	Q	日立情報システム	業務生成型	日立の DBMS ならほとんど全部 ADABAS, GIGA/E	GIGA/E と ADABAS で可	非 DBMS ファイルを利用する方式, 部品合成	機能なし
25	QMF	IBM	情報生成型	DB2, SQL/DS	あり	DBMS を利用する方式	機能なし
26	RAMIS II	Online Software International Inc.	情報生成型	IMS, DL/I, AIM, ADM, XDM, ADABAS, IDMS, MODEL 204, TOTAL	なし	RAMIS II DB と DD を利用する方式	可能
27	SAS	SAS Institute Inc.	情報生成型	IMS, DB2, ADABAS その他のメジャーな DBMS	なし	独自の RDBMS (SAS データセット) を利用する	可能
28	SOAR	ソフトウェア・エージ	情報生成型	ADABAS	なし	ADABAS に依存する方式	機能なし
29	SPEED-II	TOM 社	業務生成型	PACE, DMS, SPEED-II	SPEED-II で可	SPEED-II を利用する方式	可能
30	STYLE	フットヒル・リサーチ社 (FRI)	業務生成型	STYLE DB	STYLE DB で可	STYLE DB に依存する方式	可能
31	SUPER NATURAL	Software AG	情報生成型	ADABAS	開発中	ADABAS に依存する方式	可能
32	SYSTEM W	Comshare	情報生成型	ADABAS, DB2, SQL/DS	なし		機能なし
33	TELON	Pansophic Systems Inc.	業務生成型	IMS/DB, DB2	予定あり	VSAM を利用する方式	可能
34	TQF II	日本電気	情報生成型	RIQS II		RIQS II を利用する方式	可能
35	UFO	Online Software International Inc.	業務生成型	IMS/DB, ADABAS, TOTAL, DL/I	SQL/DB, DB2 で可	独自のファイル IOX (VSAM のシミュレート可能な可変長ファイル) と DD の利用による方式	可能
36	簡易言語の UL/204 と WORKSHOP 204 その他	Computer Corporation of America	統合化4GL	MODEL 204	予定あり	MODEL 204 に依存する方式	可能

あること.

② 豊富なデータベースとディクショナリを拠り所としたプログラム開発支援環境であること.

狭義には，さらに次の3条件を追加する立場もある[4),25)].

③ ディクショナリの完備性が保証されたシステムであること.

④ （データベースを生かした非手続き型のプログラム言語であるためには，必然的に）②のデータベースは，特に RDB でなければならない.

⑤ JCL を使わずにプログラムの開発やメンテナンスができること.

第4世代言語が上記のような定義でよいか否かは，①と⑤については 3. ②〜④については 4. で議論する.

2.2 機能モデルについて

前述したように，第4世代言語は用途もプログラム構造も異なるソフトウェア・ツールの総称であり，これを定義することが困難なために起こる混乱も少なくない．このため，アメリカの国家標準技術研究所☆ (NIST : National Institute of Standards & Technology) は，機能に着目した第4世代言語のモデル（＝機能モデル）により第4世代言語を分類・定義し，ユーザが第4世代言語を選択する場合のガイドラインを与えている[22)].

対象となるもの（ここでは第4世代言語）が提供する機能やサービスによって，対象となるものの包括的な概念を規定したものを「対象となるものの」機能モデルと呼ぶ．たとえば，OSI の参照モデルは機能モデルの一例である.

（1） 第4世代言語の機能モデル設定の目的

第4世代言語の機能モデルを設定する目的は次のとおりである．なお，このモデル自体は規格化を意識したものではないが，将来の規格化の叩き台となる公算が大である.

① 第4世代言語の基本的な定義をする.

② 第4世代言語の主な機能を定義する.

③ 第4世代言語を使ってプログラムを記述する際の共通なフレームワークを提供することによって，ユーザの教育を容易にする.

④ 市販の第4世代言語に対する客観的な分類を可能にする.

⑤ 評価や導入時の参考にする.

（2） 第4世代言語の機能モデルの設定範囲

機能モデルは，エンドユーザと専門プログラマの両者を対象にして，第4世代言語として満たすべき最少限度の仕様を規定し，さらに第4世代言語が備えてもよい，より高度な機能についても言及している．そこでは，一般的なビジネス・アプリケーション分野での第4世代言語の定義に限定している．また，第4世代言語における，第4世代言語の実現方法に関する特徴（たとえば，コンパイラ型か否かなど），生産性に関する要素（たとえば，生産性が COBOL の 10 倍以上，など），手続き型か非手続き型か，DBMS や参照言語とかいうような特徴などは機能モデルには含まない.

（3） 第4世代言語の機能モデルの構成要素

第4世代言語は，機能的にユーザ機能，データ管理機能，システム機能の三つから構成されている．詳しくは文献 22) を参照されたい.

3. ヒューマン・インタフェースからの分析

3.1 非手続き的な記述か手続き的な記述か

実際の第4世代言語を調査した結果から一般的に次のことが言える.

情報生成型 4GL では，オペレータ・コマンドのような言語によって作業指示を与えるものがほとんどである．したがって，作業が単純な場合には非手続き的な記述を実現している．しかし，作業が複雑になると，コマンドの与える順序そのものが意味をもつようになるので，非手続き的な記述にはならない.

一方，業務生成型 4GL では，データ構造を定義する部分は非手続き的な記述をほぼ実現している．しかし，データの処理仕様を記述する部分は手続き的な記述となっている．データの処理仕様に対しても非手続き的な記述を支援する 4GL は，プログラム言語としての記述能力が低いため，生成されるプログラムが定型的なものに限られる．したがって，このような言語では，複雑または大規模なプログラムをも生成できるものはほとんどない．一方，複雑または大規模なプログラムをも生成可能な 4GL のほとんどは，プログラムの記述が手続き的で，かつ，その記述レベルが（特に，データベース言語の SQL よりも）低く，COBOL で記述されたプログラムを生成するためのマクロ言語のようなものが多い.

☆ アメリカの国家標準技術研究所 (NIST) は 1988 年 8 月に改組され，それまでの国家標準局 (NBS : National Bureau of Standards) から現在の名称に変更された.

3.2 ヒューマン・インタフェース向上のための工夫

(1) WYSIWYG (=what you see is what you get) 方式

対象業務で使用する帳票や端末装置の画面などのフォーマットについては，ユーザは細かいところまで自分の意図を反映したものにしたい．第4世代言語では，このような要求に対してフォーム・ペインタ/スクリーン・ペインタと呼ばれる機能を提供することによって対処しているものが多い．これらの機能により，帳票や画面などをユーザが実際に使用するであろう，その実物と同じイメージをディスプレイ上にユーザが自らの手で作成することが可能となる．このような機能は，ユーザがディスプレイ上で見たイメージと同じものを手にすることができるという意味でWYSIWYG と呼ばれている．WYSIWYG の機能で，ユーザの意図をどこまで詳細に反映できるかという点と，フォーマット作成時の操作性は，WYSIWYG の機能を実現しているワークステーションや端末の機能によって左右される．したがって，WYSIWYG の機能を追求すれば第4世代言語のポータビリティが悪くなり，ポータビリティを追求すれば WYSIWYG の機能の実現レベルが低くなるというトレードオフの関係にある．現行の第4世代言語は，ポータビリティのほうをより重視しているので，WYSIWYG の機能の実現レベルは低い．この低さは，第4世代言語の稼動環境がワークステーションではなく，汎用機が主体になっていることとも関係している．

(2) システム操作言語の一元化

ユーティリティ機能のすべてが一つのプログラミング言語を介して有機的に結合されたシステムがある．STYLE がその例である．

この種のシステムは，システムの操作が一つの言語で統合化されているため，操作方法が思想的に統一されており，理解しやすいという利点がある．しかし，当初から計画されていなかったユーティリティ機能を追加しなければならなくなったときに，操作方法における思想を維持することが困難となるという欠点がある．

3.3 JCL が必要か否か

ヒューマン・インタフェースの面から，第4世代言語は JCL がなくてもプログラム開発やメンテナンスできるようにすべきだ．そのためには，オブジェクトコードを直接生成できるようにして，煩雑な JCL を必要とするコンパイル作業からユーザを解放しなければならないという考え方がある．このような考え方を指向したものとしては，PRO-IV, CSP, EASYTRIEVE, RAMIS II, NATURAL などの例がある．しかし，生成されたプログラムの実行効率という点では，ソースコードを生成してユーザにコンパイルさせたほうが，コンパイラのオプティマイズ機能を利用できる分だけ有利である．このような例としては LINC II, TELON などがある．

4. 第4世代言語の実現方式からの分析

本章では，第4世代言語という言語がどのような仕組みで実現されているかという視点から分類し，採用されている実現方法に起因する第4世代言語の長所や欠点について論ずる．なお，表-1 に掲げた36種類の第4世代言語のそれぞれが，その実現に以下の分析におけるどのような実現方式を採用しているか，についても表-1 に記載しておいたので併せて参照されたい．

4.1 ディクショナリの内容と完備性について

データ・ディクショナリ/ディレクトリ・システム (=DD/DS) とは，データ定義の登録，参照，抹消を行う機能である．このうち，ソフトウェア開発者やプロジェクト管理者のための管理機能を DD と呼び，マシンに対しての管理機能を DS と呼んでいる．

筆者は，DD の内容をその用途から4つに分類する．1番目は，要求定義時にユーザが与えるべき情報に欠落があった場合に，どのような情報が不足しているかを指摘するための情報である．2番目は，設計支援のための情報で，データ構造やデータ項目間の関係に関する情報などがある．3番目は，要求仕様が確定した後に，与えられた仕様を満足するプログラムを自動生成するための情報で，プログラム部品や部品の利用方法に関する情報などがある．4番目は，作成された情報システムがどのようなプログラムや部品から構成されているかを管理するための情報であり，情報システムの保守に利用する．これらの中で，第4世代言語に採り入れられているのは，2番目と4番目の DD である．なお，2番目と4番目のみを DD と呼び，これに3番目の一部（静的な情報のみ）を加えたものをエンサイクロピーディア (encyclopedia)，さらに残りを加えたものをレポジトリ (repository) と呼んで DD と区別することも多い．

1番目から3番目までのレポジトリの場合には，レポジトリとしてどのような情報をどれだけ用意すべき

かということがまだよく分かっていない．このため，これらを採り入れるには第4世代言語の対象領域をよほど絞らないかぎりその完備性を保証することは難しい．

4番目の DD の場合には，受動的 DD か能動的 DD かという概念が関係してくる．DD として登録されるが，開発中のソフトウェアに対して強制力や影響力をもたない DD を受動的 DD と呼ぶ．DD として登録されることによって初めてその情報システムで認められたデータとなり，開発中のソフトウェアに対して強制力や影響力をもつ DD を能動的 DD と呼ぶ．開発中のソフトウェアの構成要素（＝保守の単位）のすべてを対象に能動的 DD を実現しているものを，4番目の DD の意味で DD の完備性を保証された情報システムと呼ぶことがある．第4世代言語における DD の完備性という言葉は，この意味で使用されている．4番目の DD の意味での完備性を保証された第4世代言語としては，NATURAL やライフサイクル一貫支援型ツールの CANO-AID などがある．

4.2 言語構築の核となる DBMS からの分析

DBMS や非 DBMS ファイルが，第4世代言語という言語を実現するためにどのように関与しているかによって，筆者は第4世代言語を下記の三つに分類する．そして，その関与の仕方の相違から，第4世代言語のポータビリティや第4世代言語で開発可能なソフトウェアの規模などについて議論する．

4.2.1 特定の DBMS を拠り所にするもの

特定の DBMS を一つ固定し，そのエンド・ユーザ言語として開発された第4世代言語である．したがって，DBMS を言語構築の核にすることによる利点と欠点を併せもっている．たとえば，DBMS の機能を利用した大規模システムの構築に向いているという利点がある反面，DBMS を言語構築の核にしているという制約から，プログラム・ジェネレータとしての自動化率が低いという欠点がある．一方，第4世代言語のポータビリティは，その第4世代言語が言語構築の核とする DBMS のポータビリティに依存している．

DBMS にはリレーショナル型，ネットワーク型，階層型の3種類があるが，この分類に属する第4世代言語が採用している DBMS は，LINC II などの例外を除き，ほとんどがリレーショナル型のデータベース (RDB) を採用している．

（1） 情報生成型 4GL

独自の DBMS の上に実現しているものと，既存の DBMS の上に実現しているものの二つがある．前者の例としては AS, ETOILE/OP, FOCUS, MAPPER, PLANNER, RAMIS II, SAS などがあり，後者の例としては IDEAL/DQ, SOAR, SUPER NATURAL, TQF II などがある．

（2） 業務生成型 4GL

ADS/ONLINE, CASET, IDEAL/DB, IDL II, LINC II, SPEED-II, STYLE などがこの分類に属する．

LINC II は，リレーショナル型とネットワーク型の両用に使用可能な DMS II を言語構築の核として使用している．システム設計の容易さと高速処理を両立させるために，DMS II を後者として使用しているらしい．

（3） 統合化 4GL

MODEL 204 を言語構築の核とするプログラム開発環境（簡易言語 UL/204＋プロトタイピング支援ツール WORKSHOP/204 など）と NATURAL の二つがこの分類に属する．

4.2.2 特定多数の既存 DBMS を拠り所にするもの

DBMS の豊富な機能を拠り所にした第4世代言語ではあるが，言語のポータビリティを上げるために，多種類の既存 DBMS を言語構築の核として使用しており，DBMS の機能がそれらの共通部分だけになっているものが多い．このために，DBMS の機能を利用した大規模システムの構築に向くという長所も，4.2.1 の第4世代言語ほどには活かしきれていない．

（1） 情報生成型 4GL

SAS, AS, EASYTRIEVE, QMF などがこの分類に属する．

（2） 業務生成型 4GL

MANTIS などがこの分類に属する．

（3） 統合化 4GL

調査した範囲の第4世代言語には，この分類に属するものは見当たらなかった．

4.2.3 非 DBMS のファイルを拠り所にするもの

既存の DBMS 以外のファイルを拠り所にして実現している第4世代言語である．この方法は，言語構築の核として DBMS を利用していないという自由さから，プログラム・ジェネレータとしての自動化率が高くなる反面，外部ファイルとして既存の DBMS を利用できるようにしないと，大規模システムの構築が不

可能になるという欠点をもつ.

(1) 情報生成型 4GL

調査した範囲の第4世代言語には，この分類に属するものは見当たらなかった.

(2) 業務生成型 4GL

PRO-Ⅳ, TELON, CSP, UFO などがこの分類に属する.

PRO-Ⅳ は，VSAM のKSDS を内部ファイルとして言語構築の拠り所にしており，IMS (IBM)，AIM (富士通)，ADM (日立) などの DBMS を外部ファイルとして使用可能にしている.

CSP は，開発者用ライブラリ (MSL : Member Specification Library) と実行ライブラリ (ALF : Application Load File) を内部ファイルとして言語構築の拠り所にしており，既存の IBM-DBMS を外部ファイルとして使用可能にしている.

(3) 統合化 4GL

調査した範囲の第4世代言語には，この分類に属するものは見当たらなかった.

4.3 言語構築の核となる DBMS 以外からの分析

本節では，第4世代言語という言語を実現するために，どのような方式を採用しているかということについて，前節とは異なった観点から議論する.

4.3.1 情報生成型 4GL

(1) 表操作用簡易言語型

表計算用の簡易言語という形で，RDB の操作を支援するエンドユーザ向きの言語である．ユーザは DBMS を意識することなく，プログラムを開発することができる．この種の言語の成功は，適用範囲を極端に絞ることにより，言語の有効性を引き出していることにある.

ETOILE/OP, TQF Ⅱ がこの分類に属する.

(2) オフィース業務のモデル化によるもの

オフィースの業務をモデル化することによって実現している情報生成型 4GL である．MAPPER がその例である．MAPPER は，オフィースのファイル・キャビネット群の概念で表わされる独自の DB をもち，使用者はファイル・フォルダに入ったレポートを操作コマンドを使って検索・加工・編集する．レポートは，それぞれのキャビネット，抽き出し，ファイルに対応するモード，タイプ，レポートによって特定される．各レポートは，複数の行 (ライン) で構成されている.

4.3.2 業務生成型 4GL

(1) DBMS と DD/DS によるもの

DBMS と DD/DS を言語構築の拠り所にしたソースコード・ジェネレータであり，業務生成型 4GL としては最も第4世代言語らしい第4世代言語である．ADS/ONLINE, IDEAL/DF, IDL Ⅱ, LINC Ⅱ, 日本語 VAX DATATRIEVE, RAMIS Ⅱ などがこの分類に属する.

LINC Ⅱ は，大規模システムでも容易にプログラムが開発できるように，対象業務をモデル化する技法が整備されており，モデル化技法としては，RDB の構築技法である ER モデルを導入している．また，言語仕様も最初から大規模システムを開発することを前提として設計されており，言語も COBOL と同程度の記述能力をもっている.

(2) 部品合成によるもの

部品合成による方法については文献 12), 15) に詳しい解説があるので，ここでは第4世代言語に特徴的なことだけを述べる.

部品合成による方法は，部品をソースコードの形でもつものとオブジェクトコードの形でもつものとの二つに大別される．部品をソースコードの形でもつものは二つに分類できる.

一つは，データフローの概念に基づく部品を利用する方法である．ある部品の出力エリアが次に実行すべき部品の入力エリアとなるような部品間のインタフェースを設定し，レコード単位でデータの受渡しを行うような部品群を繋いでいけば，一つの実行可能なプログラムを構成できる．このような部品をデータフローの概念に基づく部品と呼ぶ.

データフローの概念に基づく部品を利用する方法は，DF-COBOL (三井造船) で初めて製品化された．その後，富士通がこの技術を買い取り，HYPER-COBOL で製品化された．第4世代言語の CASET, CSP, Q などがその例である.

もう一つは，骨組み部品 (＝プログラムの雛型) とその部分部品による方法である．骨組み部品は，プログラムの制御構造 (＝処理の流れ) に着目してその骨組みをコード化 (部品化) したもの (＝コード・スケルトン, codes skeleton) で，処理パターンとも呼ばれる．部分部品は，骨組み部品を利用して作成されるプログラムの部分を構成する部品である．したがって，プログラム作成の際には，骨組み部品の中に挿入される形で使用される．なお，骨組み部品を処理パ

ターンと呼ぶ場合には単に部品と呼ばれている．

これらの部品による方法は，第4世代言語というよりも，ライフサイクル一貫支援型のツールと呼ばれる，SEA/I（日電），EAGLE（日立），CANO-AID（キヤノンソフトウェア）などで採用されている．第4世代言語では，業務生成型4GLのIDL II，情報生成型4GLのEASYTRIEVEのように，骨組み部品のみを利用しているものが多い．

部品をオブジェクトコード（＝コンパイル・ユニット）の形でもつ方法は1種類のみで，リンケージ・テーブルを利用した動的結合型と呼ぶべきものである．ユーザの指定に応じて必要な部品を動的に結合できるようにするために，ユーザ要求の選択肢の数だけあらかじめ部品を用意しておき，動的に結合すべき部品の名称をリンケージ・テーブルの形で管理する．

この方式は，リンケージ・テーブルを生成するだけでプログラムを生成できるので，プログラム生成時間が短く，しかも，少ないメモリでシステムを構築できるという利点がある．また，部品をアセンブラで記述しておくことにより，オブジェクト効率をよくすることができる．この場合，ユーザ要求の選択肢の数だけアセンブラで部品を用意することになるので，システムのポータビリティが悪くなるという欠点がある．PRO-IVはこの方法を採用している第4世代言語であり，そこではこの欠点を補うために，外部ファイルとして既存のDBMSをサポートしている．

（3） 部品化の支援によるもの

部品合成支援よりも部品化支援に重点を置いた方法である．前者は，プログラムの作成を容易にするためには，どのような部品を提供し，それをどのように組み合わせていけばよいかを支援することに重点を置く．これに対して，後者は，どのようなソースコードを部品として定義するかの判断はユーザに委ね，ユーザの部品定義（＝部品化）と定義された部品の展開を支援することに重点を置く．そこでは，ユーザによるプログラム記述を1行でも少なくするために，部品定義のための記述を複数個集めたものをまた部品として定義すること（＝部品の階層的定義）を可能にしている．この方法の利点は，対象領域ごとにどのような部品が必要となるかということが分からなくても適用可能であり，このため対象領域によらず適用可能なことである．JASMACはこの方法の典型的な例である．

4.3.3 統合化4GL

統合化4GLは，情報生成型4GLと業務生成型4GLの機能を合わせもった第4世代言語である．それゆえ，一つで両方の機能を実現できるような仕組みが採用される．具体的には，RDBMSを言語構築の核に据え，その上の業務処理プログラムとして情報生成型4GLと業務生成型4GLを構築する方法が採用される．そこでは，DBMS自身がもつ豊富な検索機能やレポート処理機能が情報生成型4GLの構築に利用され，DBMS自身がもつオンライン処理支援機能やファイル管理機能が業務生成型4GLの構築に利用される．MODEL 204を言語構築の核とするプログラム開発環境（簡易言語UL/204+プロトタイピング支援ツールWORKSHOP/204など）とNATURALの二つがその例である．

5. プロトタイピング支援ツールとしての有効性からの分析

5.1 プロトタイピング技術の分類

筆者は，プロトタイピング技術（＝プロトタイプの作成技術）をその実現方式から次の三つに分類する．なお，プロトタイピング技術の研究動向については，文献5）に詳しい解説があるのでそれらを参照されたい．

（1） プログラミング言語によるもの

通常のプログラミング言語を使ってプロトタイプを作成する方法である．この方法には，

① プロトタイプのすべてを新規作成する．

② 既存のプログラムや部品を再利用する．

という二つの方法がある．しかし，前者は，短期間に少ない工数で作成するという点では問題がある．後者は，再利用を可能にするためにはどのようなプログラムや部品をどれだけ準備すればよいか，という問題を解決しないかぎりにおいては有効な方法とは認めにくい．

（2） 実行可能な仕様記述によるもの

要求仕様を記述し，これをコンパイルするだけで，プログラムとして実行可能であれば，ライフサイクルの早い段階にプログラムを短期間に低コストで作成することができる．したがって，このような仕様記述言語とその処理系があれば，プロトタイピング技術として利用できる．このような技術を実行可能な仕様記述（executable specification）という．実行可能な仕様記述の実現方式については，文献13），27）に詳しい解説があるのでそれらを参照されたい．

（3） アニメーションによるもの

対象とするシステムの振舞いをグラフや図表によっ

て動的に表現できれば，プロトタイピング技術として利用できる．要求仕様はインタラクティブに与えられるのが普通で，その実行にはコンパイルを必要としないという特徴がある．このような方法をアニメーションによるプロトタイピング[16]という．

筆者は，アニメーションによるプロトタイピングの実現方法を次の二つに分類する[16]．

① WYSIWYG 方式

ユーザが画面を通じて確認したもの（＝プロトタイプ）が，そのままユーザが得られるもの（＝最終システム）になるという方法である．このような方法をWYSIWYG 方式によるプロトタイピングと呼ぶ．対象システムの叩き台として，これ以上確実なプロトタイピングはない．WYSIWYG が可能なのは，対象システムの中でビジュアルなもの（＝対象システムの入出力物）を動作させることが，そのまま対象システムの振舞いを確認することになる場合である．このような入出力物は，事務処理プログラムにおける画面や帳票などに限られる．したがって，WYSIWYG 方式によるプロトタイピングとは，対象システムで使用する画面や帳票などをワークステーションや端末の画面上に作成し，それを動作させることによって，その形式や使い方を確認する方法である．

この方法をプロトタイピングに適用した研究事例としては PAPS[16] などがある．

② 等価回路方式

システムの振舞いを規定する構成要素と理論的に等価な図形やグラフを使い，それを動作させることによって，対象システムの振舞いを確認する方法である．このような方法を等価回路方式によるプロトタイピングと呼ぶ．このような方法が可能な場合は二つある．一つは，対象システムの振舞いが状態遷移で規定される場合であり，状態遷移をグラフの遷移で表す方法である．STATEMATE[7] は，この方法をプロトタイピングに適用した好例である．もう一つは，対象業務の振舞いがデータフロー図やバブルチャートなどで規定される場合であり，業務フロー図などの上をデータやものが移動していくさまを図示する方法である．VIP[1] がこの方法の有効性を示唆している．

第4世代言語は仕様記述言語とはいいがたいが，プロトタイピング支援環境としては，「実行可能な仕様記述」を指向したプログラミング言語だとみなすことができる．仕様記述言語とプログラミング言語の区別はあまり明確ではないが，概念的には次のように区別

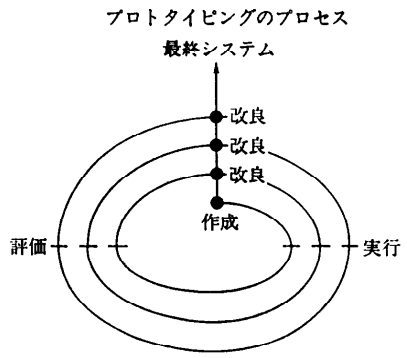

図-1 プロトタイピングを繰り返すことによるソフトウェア開発

することができる．すなわち，その言語で記述されたものが，主として操作的な意味をもつものがプログラミング言語であり，宣言的な意味をもつものが仕様記述言語である．この意味では，実行可能な仕様記述言語は，宣言的な意味と同時に操作的な意味をもつ言語だということができる．

第4世代言語がプロトタイピング支援ツールとして使用可能である最大の理由は，ソフトウェア開発におけるライフサイクルのターンアラウンド・タイムが短いことにある．したがって，第4世代言語を利用すれば，ソフトウェアを試行錯誤で開発することができる．このような開発手法は，ソフトウェア・ライフサイクルとしては，プロトタイピングを繰り返すことによるソフトウェア開発（**図-1 参照**）とみなせる．ここでは，プロトタイプの完成がそのまま最終的ソフトウェアの完成となるところにその特徴がある．

5.2 プロトタイピング支援ツールとしての有効性

ソフトウェア生産ツールが，プロトタイピング支援ツールとして有効であるための条件は

① 与えられた仕様を忠実に表現する動的なモデルを，短い工期かつ少ない工数で実現できること．

② 一度作成したモデルの修正が容易であること．

③ 対象とするプログラムの範囲が十分広く，さまざまな要求を仕様として表現できること．

④ 要求仕様の与え方が容易であること．

⑤ 要求仕様の与え方そのものの習得が容易であること．

⑥ 作成済みのプロトタイプから，最終システムを速やかに作成する手段をもつこと．

の6つである[16]．

第4世代言語は，ソフトウェア工学的な完成度は高

くないが，上記の条件①～⑥をすべて満足している．したがって，この点から第4世代言語がプロトタイピング支援ツールとして有効であることが裏付けられる．そして，現状の第4世代言語がプロトタイピングに特に有効である局面としては，次の三つがあげられる．

◆試行錯誤的にプログラミングする場合

◆画面・帳票などのレイアウトを設計する場合

これらのレイアウトを強力に支援するには，スクリーン・ペインタやフォーム・ペインタの採用が必須である．

◆環境使い分け型により開発する場合

第4世代言語の中には，開発環境と運用環境を明確に区別しているものがある．「環境使い分け型」と呼ぶべき第4世代言語がそれである．これらは，プロトタイプの作成を開発環境で行い，できたプロトタイプを運用環境に合うように変換することにより，最終的ソフトウェアを作成するものである．この種の第4世代言語では次の二つが有効であろう．一つは，CASETのように，ワークステーションを開発環境とし，ホストを運用環境として位置づける方法である．もう一つは，CSPのように，作成されたプロトタイプを変換により，開発環境とは異なるOS上でも実行可能とする方法である．

6. 適用範囲や記述能力からの分析

（1） 第4世代言語の適用範囲

システム開発の対象となるソフトウェアは，システムの振舞いが何によって決定するかということから，次のように分類できる[16]．

① reactive software

event-driven な振舞いをするソフトウェアのことで，システムの振舞いがシステムの内部状態とそれへの制御信号によって決定するという特徴がある．

② transformational software

data-driven な振舞いをするソフトウェアのことで，システムの振舞いがデータの入出力仕様とその変換仕様によって決定するという特徴がある．したがって，データ構造が決まれば，これを処理するプログラムの構造も決定しやすいという特徴がある．

③ time-serial software

time-driven な振舞いをするソフトウェアのことで，システムの振舞いが時間経過によって決定するという特徴がある．

④ functional software

システムの対象が①～③のいずれかに属する機能の集合とみなされるソフトウェアのことで，機能間に流れるデータがあるという観点から，システムの振舞いがデータフロー図やバブルチャートなどで規定されるという特徴がある．

第4世代言語が対象とするソフトウェアの範囲は，一部（SASなど）の例外を除けば，上記の分類における transformational software に限られている．

（2） 第4世代言語の記述能力について

言語の記述が手続き的か非手続き的かによって，プログラミング言語の記述能力にどう影響しているか，ということについては**3.1**で議論した．ここでは，記述能力の大きさをコマンド機能の大きさとの関係で議論する．プログラミング言語の記述能力はコマンド機能の大きさに反比例する．すなわち，コマンド機能が大きくなれば記述能力は逆に小さくなる．一方，プログラムの生産性はコマンド機能の大きさに比例する．したがって，言語の記述能力とプログラムの生産性はトレードオフの関係にある．

理想的には，言語の記述能力を確保しつつ，プログラムの生産性を高くすることであるが，この相矛盾する要求を第4世代言語ではどう解決しているのであろうか．それは，第4世代言語の適用範囲を transformational software に絞っていることと関係する．transformational software は，データ構造が決まればこれを処理するプログラムの構造も決定しやすいという特徴をもっている．このため，データ構造の定義（非手続き的な記述が可能）とデータを操作する機能（検索，編集，更新など）については，木目の細かな記述を可能にし，エラー処理やその他の機能についてはコマンド機能を大きく設定するということでこの矛盾を解決している．

7. ユーザによる第4世代言語の評価

本章では，第4世代言語を実際に使用しているユーザが第4世代言語をどう評価しているかを，情報処理振興事業協会が実施したアンケート調査とヒアリング調査の結果[2]（**表-2**）を基に紹介する．

（1） 記述能力について

プログラミング言語の記述能力はコマンド機能の大きさに反比例する．また，あまり正確ではないが，コマンド機能の大きさでその記述が非手続き的か否かを判断できないこともない．このような観点から，コマ

表-2　第4世代言語を実際に使用しているユーザによる第4世代言語の評価

項番	4GLの評価項目	評価項目に対する回答		情報生成型4GL	業務生成型4GL	統合化4GL	4GL全体
1	コマンド機能の大きさ(記述能力)	小さい(手続き的)		8%	39%	33%	26%
		ちょうどよい		62%	44%	67%	57%
		大きい		31%	17%	0%	17%
2	エンドユーザだけでソフトウェア開発可能か	EDP部門のみ		—	—	—	33%
		EDP部門から外注		—	—	—	28%
		エンドユーザも参加		—	—	—	37%
		ベンダに開発依頼		—	—	—	2%
3	言語として使い易いか	エンドユーザにとって	使い易い	46%	7%	50%	33%
			普通	31%	26%	17%	23%
			使い難い	23%	67%	33%	44%
		EDP部門の人にとって	使い易い	64%	61%	83%	64%
			普通	29%	39%	17%	33%
			使い難い	7%	0%	0%	3%
4	修復期間	2～3日でよい		—	—	—	35%
		1週間程度でよい		—	—	—	38%
		2週間程度でよい		—	—	—	7%
		1カ月程度でよい		—	—	—	10%
		2カ月程度でよい		—	—	—	10%
5	ソフトウェア開発に占める4GLによる開発の比率	一つの4GLだけで開発	～5%	—	—	—	36%
			～10%	—	—	—	12%
			～20%	—	—	—	6%
			～40%	—	—	—	21%
			～60%	—	—	—	3%
			～80%	—	—	—	9%
			～100%	—	—	—	12%
			平均	—	—	—	32%
6	既存のシステムとの連結	既存のプログラムとの連結	容易	40%	25%	50%	32%
			可能	30%	50%	34%	41%
			不可能	30%	25%	16%	27%
		既存のファイルとの連結	容易	83%	36%	57%	54%
			可能	17%	45%	28%	28%
			不可能	0%	19%	18%	18%
		既存のDBMSとの連結	容易	27%	5%	33%	17%
			可能	46%	68%	33%	51%
			不可能	27%	27%	33%	32%

項番	4GLの評価項目	評価項目に対する回答			情報生成型4GL	業務生成型4GL	統合化4GL	4GL全体
7	構造プログラミングに忠実か	忠実である			50%	84%	83%	74%
		忠実でない			50%	16%	17%	26%
8	プログラムのデバッグは容易か	容易			64%	67%	100%	71%
		普通			36%	29%	0%	27%
		困難			0%	4%	0%	3%
9	他人の作ったプログラムの保守	保守できる			100%	93%	100%	98%
		保守できない			0%	7%	0%	2%
10	プロトタイプを素早く作成することは容易か	容易			73%	61%	83%	68%
		普通			18%	33%	17%	26%
		困難			9%	6%	0%	6%
11	生産性はCOBOLなどの言語と比べて何倍になったか	開発作業	1～2倍		—	—	—	43%
			～3倍		—	—	—	27%
			～4倍		—	—	—	3%
			～5倍		—	—	—	14%
			～10倍		—	—	—	10%
			10倍を超える		—	—	—	3%
			平均		—	—	—	3.3倍
		保守作業	1～2倍		—	—	—	17%
			～3倍		—	—	—	7%
			～4倍		—	—	—	0%
			～5倍		—	—	—	7%
			～10倍		—	—	—	7%
			10倍を超える		—	—	—	7%
			平均		—	—	—	3.7倍
12	コンピュータの負荷はことどとまでてなった比べうか	CPU	変わらない		46%	39%	38%	41%
			やや負担		38%	50%	62%	49%
			非常に負担		16%	11%	0%	10%
		メモリ	変わらない		71%	20%	50%	37%
			やや負担		29%	67%	50%	56%
			非常に負担		0%	13%	0%	7%

ンド機能の大きさを小さい（手続き的）/ちょうど良い/大きい（非手続き的）の三者択一で回答させた．このうちちょうど良いと答えたユーザは，情報生成型4GLでは62%，業務生成型4GLでは44%，統合化4GLでは67%，4GL全体では57%となっており，業務生成型4GLを除けば，コマンド機能の大きさはちょうど良いと評価しているようである．

（2）　エンドユーザだけでソフトウェア開発可能か

第4世代言語を用いれば，エンドユーザだけでソフトウェア開発可能か否かという問題については，98%がノー（内訳はEDP部門のみが33%，ソフトウェア・ハウスへ外注が28%，エンドユーザもタッチしたが37%）と答えている．

（3） 使いやすいか

この問いに対して容易/普通/困難の三者択一で回答させた．その結果，エンドユーザは業務生成型 4GL では 67%，4GL 全体でも 44% が使いにくいと答えており，エンドユーザにとっては，第4世代言語（それも特に業務生成型 4GL）は使いやすいとは言えないようである．一方，EDP 部門は情報生成型 4GL では 64%，業務生成型 4GL では 61%，統合化 4GL では 83%，4GL 全体では 64% が使いやすいと答えており，EDP 部門は第4世代言語が使いやすいと評価している．

（4） 修得期間は短くてよいか

修得期間は，2～3日でよいが 35%，1週間程度でよいが 38% となっており，基本的なコーディングや操作については短期間でよいという評価である．

（5） 第4世代言語による開発の新規開発に占める比率

一つの第4世代言語で開発するソフトウェアの，新規開発ソフトウェア全体に占める比率は，5% 以下が 36% と最も多く，21～40% が 21%，6～10% と 81～100% が 12% の順になっており，平均では 32% となっているが回答にばらつきが見られる．その理由はユーザによって，第4世代言語の使途に対する考え方や開発要求のあるソフトウェアにばらつきがあるからだと推測される．

（6） 既存のシステムとの連結は容易か

この問いに対して，容易/可能/不可能の三者択一で回答させた．その結果，容易または可能と答えたユーザが，4GL 全体で，既存のプログラムとの連結では 73%，既存のファイルとの連結では 72%，既存の DBMS との連結では 68% となっており，多くの場合，既存のシステムとの連結は可能だという評価である．

（7） 構造化プログラミングに忠実か

この問いに対して，忠実である/忠実でないの二者択一で回答させた．このうち忠実であると答えたユーザは，情報生成型 4GL では 50%，業務生成型 4GL では 84%，統合化 4GL では 83%，4GL 全体では 74% となっており，情報生成型 4GL を除けばおおむね構造化プログラミングに忠実だという評価である．

（8） デバッグは容易か

この問いに対して容易/普通/困難の三者択一で回答させた．このうち容易であると答えたユーザは，情報生成型 4GL では 64%，業務生成型 4GL では 67%，統合化 4GL では 100%，4GL 全体では 71% となっており，COBOL などの言語に比べて容易だという評価である．

（9） 他人の作ったプログラムを保守できるか

この問いに対して可能/不可能の二者択一で回答させた．このうち可能であると答えたユーザは，情報生成型 4GL では 100%，業務生成型 4GL では 93%，統合化 4GL では 100%，4GL 全体では 98% となっており，可能だという評価である．

(10) プロトタイプの作成/修正は容易か否か

この問いに対して容易/普通/困難の三者択一で回答させた．このうち容易であると答えたユーザは，情報生成型 4GL では 73%，業務生成型 4GL では 61%，統合化 4GL では 83%，4GL 全体では 68% となっており，おおむね容易だという評価である．

(11) 生産性について

ベンダやメーカによれば，COBOL などの言語に比べて生産性が 3～10 倍は向上するといわれているが，ユーザはどう評価しているか．このことについては，開発作業では平均 3.3%，保守作業では平均 3.7% となっている．

ただし，上記の数値が，プログラミング工程だけでみた回答か，ライフサイクル全体でみた回答か判然としないところがあるが，「コーディングでは2倍，設計工程を含めれば 1.3 倍」などの回答があるように，大半はプログラミング工程に限って評価して回答しているようである．

(12) コンピュータの負荷はこれまでと比べてどうか

この問いに対して，変わらない/やや負担/非常に負担の三者択一で回答させた．その結果，CPU については 4GL 全体で，41% が変わらない，49% がやや負担と答えており，変わってもやや負担が増える程度となっている．メモリについては 4GL 全体で，37% が変わらない，56% がやや負担と答えており，やや負担が増加していると評価している．

(13) ソフトウェア開発の体制や工程を変化させたか

第4世代言語の導入により，ソフトウェアの開発体制やライフサイクルに変化が生じているか，という問いに対しては，総じて従来と変化なしという回答であった．しかし，第4世代言語の導入によりプロトタイピングによるソフトウェア開発が浸透しつつあるといえるであろう．

8. おわりに

第4世代言語について，①第4世代言語の定義とその機能モデル，②マンマシン・インタフェースからの分析，③第4世代言語の実現方式からの分析，④プロトタイピング支援ツールとしての有効性からの分析，⑤適用範囲や記述能力からの分析，⑥ユーザによる第4世代言語の評価の6点から分析し解説した．本稿よりも詳しい内容については文献2) と26) を参照されたい．最後に，以上の分析から得られた，第4世代言語の将来方向について論じたい．

（1） 帳票の知的自動認識機能をもつ第4世代言語

ユーザが第4世代言語を使うときには，自分が要求する情報やプログラムがどのようなものであるかを指定するための情報（＝要求定義情報）を第4世代言語で必ず与えなければならない．この作業がだれにでもできるようにすることが第4世代言語の理想である．それはエンドユーザだけでソフトウェア開発できるようにすることである．それには二つの方向がある．一つは要求定義作業の軽減である．もう一つは要求定義作業の容易化である．

要求定義作業の極端な軽減という方向で第4世代言語の将来方向を示したものが BELIEVE[31] である．BELIEVE は，エンドユーザだけでソフトウェア開発できるようにするために，次のような機能を支援している．

① イメージリーダから読み取るだけで，帳票や伝票の形式を自動的に認識して，対応する画面情報などを自動生成する．

② 伝票・伝票の形式やそこで使われている用語から，どこでどのように計算するかを自動的に認識して，対応する計算処理プログラムを自動生成する．

（2） プロトタイピング指向の第4世代言語

第4世代言語がソフトウェア生産ツールとして強力なのは，試行錯誤による開発が可能な生産ツールだからである．したがって，プロトタイピング指向の第4世代言語が有効である．プロトタイピング作業をより容易にするためには，transformational software の性質に着目したプロトタイピング手法（＝アニメーションによる方法）[16]が強力である．

（3） 自動生成型の第4世代言語

ソフトウェアの生産性を向上させるには，ユーザによるデバッグ作業を軽減させることが不可欠である．そのためにはプログラム自動生成型の第4世代言語が有効である．将来的には，完全自動によるプログラム生成により，ユーザをデバッグ作業から解放することが理想である[11),14),17)]．

（4） 開発環境と実行環境を区別した第4世代言語

ワークステーションがもつ強力な WYSIWYG 機能を利用してソフトウェアを開発し，ターゲット・マシンの上で実行するような第4世代言語が有効である．

（5） SQL に準拠した DBMS を核にした第4世代言語

DBMS は SQL を標準でサポートするのが国際的な趨勢である．したがって，第4世代言語においても SQL を標準でサポートするものが生き残るであろう．

（6） 分散処理を可能にする第4世代言語

作成するソフトウェアの規模が大きくなると，1台の集中型システムでは処理できなくなることがある．このような事態に対処するために，コンピュータを複数箇所に設置し，これらを通信回線で結合した分散処理システムにする必要がある．したがって，分散処理機能を支援する DBMS を言語構築の核に据えた，分散処理可能な第4世代言語が生き残るであろう．

謝辞 調査（インタビュー調査，資料提供，試用調査）にご協力いただいた第4世代言語のベンダの方々および三菱総合研究所の西山聰氏にお礼を申しあげたい．また，表-1 の一部と 6. の内容は，情報処理振興事業協会の山崎明研究員の調査報告書[2)] に負うところが大きい．特にお礼を申しあげたい．

参考文献

1) 阿草清滋他：上流工程支援ツール VIP の試作，日本ソフトウェア科学会第 6 回大会論文集，pp. 177-180 (1989).
2) 「第4世代言語（4GL）の動向調査及び Σ システムでの利用に関する調査研究」に関する報告書，情報処理振興事業協会 (1988).
3) 池田秀人：第4世代言語の現状と将来，情報処理学会「アドバンスデータベースシステム」シンポジウム論文集，pp. 53-62 (Dec. 1988).
4) 石井義興：近代化への挑戦 なぜ第4世代言語か，Cmoputer Report, 1985年10月臨時増刊号，pp. 26-32 (1985).
5) 伊藤 潔，本位田真一：プロトタイピング支援ツール，情報処理，Vol. 30, No. 4, pp. 387-395 (1989).
6) 月刊コンピュータ編：ソフトウェア革命第4世代言語，コンピュータエイジ社 (1988).
7) Harel, D., Lachover, H., Naamad, A., Politi, M., Sheraman, R. and Shtul-Trauring, A.:

STATEMATE: A Working Enviroment for the Development of Complex Reactive Systems, Proc. of the Tenth IEEE International Conference on Software Engineering (Singapore, Apr. 13-15), IEEE Press, New York (1988).
8) 広がるソフトウェア革命，日経コンピュータ別冊 (10 Sept. 1986).
9) 角井正昭ほか：ソフトウェア開発変換システム，事務管理，Vol. 25, No. 11, pp. 103-159 (1986).
10) 栗田昭平：コンピュータ技術最前線をゆく，7「第4世代言語の普及が静かに進行」，bit, Vol. 16, No. 9, pp. 1116-1125 (1984).
11) 古宮誠一：部品合成によるプログラム自動合成システム PAPS～知識工学的アプローチを用いたその実現方式，情報処理学会研究報告，87-SF-21, Vol. 87, No. 40, pp. 5-12 (1987).
12) 古宮誠一，原田 実：部品合成による自動プログラミング，情報処理，Vol. 28, No. 10, pp. 1329-1345 (1987).
13) 古宮誠一：プロトタイピング技術の分析と実用化への方向付け，情報処理振興事業協会技術センター第6回技術発表会論文集，pp. 37-46 (1988).
14) 古宮誠一：自動プログラミングシステムで生成されるプログラムの品質保証方法，電子情報通信学会技術研究報告，SS 89-7 (July 1989).
15) Komiya, S.: Automatic Programming System by Composing Program Components and Its Realization Method, Future Generation Systems 5, North-Holland, pp. 151-161 (1989).
16) 古宮誠一：アニメーションによるプロトタイピング～制約を利用したその実現方式～，情報処理学会研究報告，SE 71-16, pp. 52-65 (Feb. 1990).
17) 古宮誠一：性能予測機能を持つ自動プログラミングシステム，情報処理学会研究報告，SE-73-10, pp. 75-82 (July 1990).
18) Martin, J.: Application Development Without Programmers, Prentice-Hall (1982).
19) Martin, J.: Fourth-Generation Languages, Vol. 1, 2, and 3, Prentice-Hall (1985).
20) 日経データプロ・ソフト：ソフトウェア開発支援ツール各論，NS 2-011-001～110 (June 1986).
21) 日経データプロ・ソフト：ソフトウェア開発支援ツール各論（第2部），NS 2-101-201～292 (Sept. 1988).
22) 日経データプロ・ソフト：第4世代言語の機能モデル，NS 2-106-001～009 (Apr. 1984).
23) 日経データプロ・ソフト：米国における第4世代言語の選択基準，NS 2-107-001～011 (Sept. 1988).
24) 日経データプロ・ソフト：レビュー，NS 2-151-151～505 (July 1987).
25) ピータ・パジェ (Peter Page)：対談 第4世代言語の方向性を語る，Computer Report, 1986年4月号，pp. 66-71 (1986).
26) プロトタイピング・ツールの有効性と効果的な利用方法の調査～プロトタイピング・ツールとして第4世代言語を使用した場合～，情報処理振興事業協会 (1988).
27) 佐伯元司：実行可能な仕様記述，情報処理，Vol. 28, No. 10, pp. 1346-1358 (1987).
28) 末舛史郎ほか：第4世代言語 NATURAL を用いたプロトタイピング用インタフェース KAPRI について，情報処理学会研究会報告データベース・システム，49-4 (20 Sept. 1985).
29) 寺田賢二，大古場進ほか：特集 第4世代言語の選び方，使い方，事務管理，Vol. 26, No. 3, pp. 15-105 (1987).
30) 特集 生産性向上ツール，コンピュータレポート，1987年2月号，pp. 33-73 (1987).
31) 野口耕平ほか：オペレーティングシステム "MIOS 7/AS" の基本技術，日立評論，Vol. 71, No. 11, pp. 57-64 (1989).
32) 山崎 明：第4世代言語 (4 GL) の動向調査および Σ システムでの利用に関する研究，技術センター第7回発表論文集，情報処理振興事業協会，pp. 179-185 (1988).
33) Yankee Group: Exploting Software Technologies (Dec. 1986).

参考文献として，上記の他にマニュアル類があるが，多くなるので記載を省略した．

（平成2年2月8日受付）