ソフトウェア工学 64－7
（１９８９．２．２）

ソフトウェア工学的観点による第４世代言語の分析と評価

古宮　誠一
情報処理振興事業協会　技術センター

最近、プログラムの開発現場では、第４世代言語（４ＧＬ）に期待を寄せる人が多い。そして、実際に使ってみると、使い易く生産性も向上したというユーザも多い。ところが、４ＧＬ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、開発者による論文等の報告は殆どない。しかも、その殆どが商用化されているので、その実現技術に至っては全く明らかにされていない。そこで、情報処理振興事業協会では、「４ＧＬのプロトタイピング・ツールとしての有効性」について調査した。この論文では、ソフトウェア工学的観点から４ＧＬを分析し評価するとともに、４ＧＬの将来像とも言うべき試作ツールの構想を明らかにしている。

An Analysis and Evaluation of 4th Generation Language from the Viewpoint of Software Engneering

Seiichi KOMIYA

Software Technology Center
Information-technology Promotion Agency, Japan (I.P.A.)

6F Shuwashibakoen 3-chome BLG,
1-38, Shibakoen 3-chome, Minatoku, Tokyo 105, Japan (I.P.A.)

Recently, there are many programmers and software users who lay their hope on 4th generation languages (4GLs). Experiments in using 4GLs actually, suggest that we can use 4GLs easily and increase the productivity of software by the use of them. But, there are almost no papers on 4GLs written by developers, because 4GLs were not developed by the researchers, but were produced at the request of programmers and software users. Moreover, their realization method remain unexplained, because almost all of them are merchandized. Therefore, I.P.A. investigated efficency of using a 4GL for prototyping. This paper describes an analysis and evaluatin of 4GLs from the viewpoint of software engneering, and describes an idea of the future 4GL which the author of this paper et al. are developing.

1．はじめに

（1）4GLの調査とその動機

最近、プログラムの開発現場では、4GLに期待を寄せる人が多い。そして、実際に使ってみると、使い易く生産性も向上したというユーザも多い。しかも、ソフトウェアを試行錯誤で開発できるという。とすれば、4GLをプロトタイピング・ツールと見なすこともできる筈である。ところが、4GL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、論文等による開発者の報告は殆どない。しかも、その殆どが商用化されているので、その実現技術にいたっては全く明らかにされていない。

そこで、IPAの「第四世代言語（４ＧＬ）の動向調査およびΣシステムでの利用に関する調査研究」という研究プロジェクトでは、4GLの利用の立場から下記の20社33種類のパッケージを採り上げ、ヒアリングによる調査をした。

ETOILE/OP(日立),ADS/ONLINE(Cullent Software),ANSWER/DB(ｽﾀｰﾘﾝｸﾞｿﾌﾄｳｪｱ),MARK Ⅳ(同左),AS(IBM),QMF(IBM),CANO-AID(ｷﾔﾉｿﾌﾄｳｪｱ),日本語VAX DATATRIEVE(DEC),TELON(Pansophic Systems),EASYTRIVE PLUS(同左),FOCUS(Information Builders),IDEAL/DF(Applid Data Research),IDEAL/DQ(同左),UFO(Onlile Software International),RAMIS Ⅱ(同左),IDLⅡ(日電),JASMAC(日本ｼｽﾃﾑｻｲｴﾝｽ),JASPL(同左),LINCⅡ(ユニシス),MAPPER(同左),MANTIS(Cincom Systems),NATURAL(Software AG),SUPER NATURAL(同左),SOAR(ｿﾌﾄｳｪｱ･ｴｰｼﾞｰ),SYSTEM W(同左),PRO-Ⅳ(Pro Computer Services),Q(日本ﾋﾞｼﾞﾈｽｺﾝｻﾙﾀﾝﾄ),SAS(SAS Institute),SPEED Ⅱ(TOM社),STYLE(FRI社),UL/204(Computer COrporation of America),PLANNER(富士通),CASET(同左)

（2）調査対象と調査方法

また、IPAには「ソフトウェア・プロトタイピング技術の調査研究」という別の研究プロジェクトがあり、「第４世代言語をプロトタイピング・ツールとして使用した場合の有効性とその効果的な利用方法」について調査することにした。

調査の対象を4GLの中でも、特にコンパイル（逐次コンパイルを含む）するだけでそのまま直接実行可能なものだけに絞った。そしてそのようなもの中のから、さらにAS(IBM),TQFⅡ(日電),CASET(富士通),CSP(IBM),EASYTRIEVE(Pansophic Systems),ETOILE/OP(日立),LINCⅡ(ユニシス),NATURAL(Software A.G.),PRO-Ⅳ(Pro Computer Services),SAS(SAS Software),STYLE(Foothill Research Inc.,CSO),UL/204(Computer Corporation of America)の12個を選び、文献調査を行なった。なお、この中のNATURAL,STYLE,UL/204の３つについては、文献調査だけでは不足する情報を補い、詳細な分析に必要な情報を収拾するために、さらにインタビュー調査を行なった。

（3）機能による調査対象の分類

今回調査を行なった１２種類の4GLを機能面から分類すれば、図１と図２のようになる。

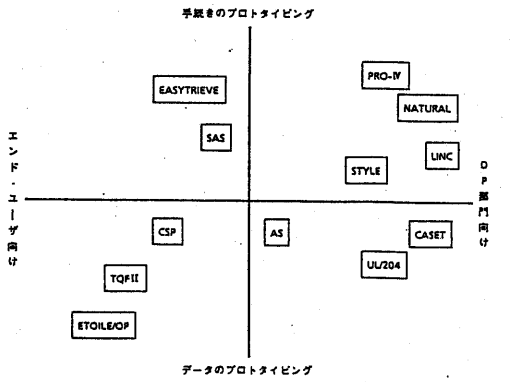


図１　ﾌﾟﾛﾄﾀｲﾋﾟﾝｸﾞの対象とﾕｰｻﾞによる分類

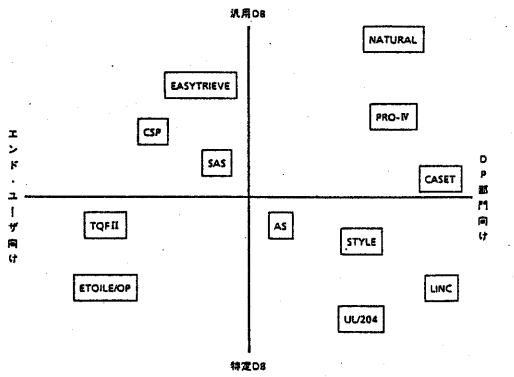


図２　ﾃﾞｰﾀﾍﾞｰｽの種類とﾕｰｻﾞによる分類

2．4GLの定義とその考察

4GLを定義することは大きな困難を伴う。それは、4GLが非常に多岐に亘ったソフトウェア・ツール、それも、用途もプログラム構造も異なるツールの総称だからである。4GLは、その用途から「情報の生成／検索／編集用」のインフォメーション・ジェネレータ、「プログラム開発用」のアプリケーション・ジェネレータ、これらの機能を合わせ持った統合化4GL の３つに大別できる。調査対象の 4GLをこの分類に当てはめてみると次のようになる。インフォメーション・ジェネレータに属するものは AS, EASYTRIEVE, SASの３つである。統合化4GL に属するものは NATURAL, UL/204、

CSPの３つである。その他のものは、すべてアプリケーション・ジェネレータに属する。

これらの分類の中で我々に興味があるのは、アプリケーション・ジェネレータと統合化4GL のプログラム開発機能である。ここで、これらを「プログラム開発用の 4GL」と呼ぶことにすると、「プログラム開発用の 4GL」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あること。

②豊富なデータベースとディクショナリを拠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[3),18)]。

③ディクショナリが完備されていること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を使わずにコンパイルできるプログラム言語であること。

しかし、よく調べてみると、①を満足するものはプログラム言語としての記述能力が低く、従って、これによって生成されるプログラムも定型的なものに限られる。従って、複雑または大規模なプログラムをも生成できるものは殆どない。一方、複雑または大規模なプログラムをも生成可能な 4GLの殆どは、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。また、③については、ディクショナリをどこまで用意すべきかということが、まだよく判っていないので、ツールの対象領域をよほど絞らない限り完備性を満足することは難しい。従って、実用レベルのツールで③を満足するものは恐らく皆無だと思われる。

3．システムの実現方式からの分析

4GLは、システムの実現にDBMSなどがどのように関与しているかということによって下記の３つに分類できる。ここで、この３つの分類に共通して言えることは、システムのポータビリティを追求すれば、高機能ワークステーションの機能を活かした機能のサポートがおろそかになるという傾向があるということである。

3．1　特定のDBMSを拠所にするもの

特定のDBMSを１つ固定し、そのエンド・ユーザ言語として開発された 4GLである。従って、DBMSをシステムの核としていることによる利点と欠点を併せ持っている。例えば、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をシステムの核としているという制約から、アプリケーション・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、 4GLのシステムとしてのポータビリティは、その 4GLが核とするDBMSのポータビリティに依存している。今後期待される方向としては、ユーザに分散処理システムとしてのロケーションを意識させずに、分散処理用アプリケーション・プログラムの開発を支援する機能であろう。

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 4GLが採用しているDBMSは、LINCⅡなどの例外を除き、殆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）インフォメーション・ジェネレータ

①表操作型（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。この種のシステムの成功は、適用範囲を極端に絞ることにより、この種の言語の有効性を引き出していることにある。

ETOILE/OP(RDBFを核とする),TQFⅡ(RIQSⅡを核とする)がこの分類に属する。

②総合システム型

RDBMS の持つ利点を最大限に活かして作成された総合的なインフォメーション・ジェネレータである。

MAPPER(MAPPER･DBMSを核とする)、IDEAL/DQ(IDEAL/DBを核とする)、 QMF(DB2やSQL/DSを核とする)、SOAR(ADABASを核とする)、 PLANNER(独自のDBMS)などがこの分類に属する。

（２）アプリケーション・ジェネレータ

①部品合成型

プログラム部品を利用可能にするために RDBを持ち、これを拠所にしてシステムを実現しているものである。CASET(独自のRDBMSを核とする)やIDLⅡ(RIQSを核とする)がこの分類に属する。

②言語による統合システム型

ユーティリティ機能のすべてが１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である。この種のシステムは、システムの操作が１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、理解し易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。

STYLE(STYLE DBMSを核とする)がこの分類に属する。

③業務モデル作成型

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いる 4GLである。このような 4GLの例としてはLINCⅡがある。LINCⅡは、モデル化技法として RDBの構築技法であるERモデルを導入している。システムの核としては、リレーショナル型とネットワーク型の両方に使用可能な DMSⅡを使用している。システム設計の容易さと高速処理を両立させるために、 DMSⅡを後者として使用しているらしい。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持っている。

(3)統合化4GL

RDBMS の豊富な機能を拠所にして統合化4GL の機能を実現するもので、 NATURAL(ADABASを核とする)と MODEL 204をシステムの核とするプログラム開発環境(簡易言語UL/204 + ﾌﾟﾛﾄﾀｲﾋﾟﾝｸﾞ・ﾂｰﾙWORKSHOP/204等)がこの分類に属する。

3．2　不特定多数の既存DBMSを拠所にするもの

DBMSの豊富な機能を拠所にしたシステムではあるが、システムのポータビリティを上げるために、システムが拠所にするDBMSとして、多種類の既存DBMSの使用を可能にするものである。そして、このために、DBMSの機能がそれらの共通部分だけになっているとすれば、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、3．1の 4GLほどには活かしきれていないということになる。

(1)インフォメーション・ジェネレータ

プログラム開発よりも、与えられたデータの処理や分析を目的として利用するものである。良い結果が得られるまでは試行錯誤的に処理を行なうが、目的が達成された後は、メンテナンスをあまり必要としないプログラムの作成に利用される。これらは、対象とするアプリケーションによって次のように分類できる。

①専門家用

SASがこの分類に属する。

② DSS用

ASがこの分類に属する。

③インフォメーション・センター用

EASYTRIEVE, FOCUS, RAMISⅡなどがこの分類に属する。

(2)アプリケーション・ジェネレータ

調査した範囲の 4GLには、この分類に属するものは見当たらなかった。

(3)統合化4GL

既存のDBMSユーザを対象に 統合化4GLを構築するものである。この分類に属する 4GLは、 統合化4GLと言っても、主体とするアプリケーション・ジェネレータ機能に、インフォメーション・ジェネレータとして一部の機能（検索してレポートを作成する機能）を追加した程度のものである。MANTISなどがこの分類に属する。

3．3　非DBMSのファイルを拠所にするもの

既存のDBMS以外のファイルを拠所にしてシステムを実現しているものである。この方法は、システムの核としてDBMSを利用していないという自由さから、アプリケーション・ジェネレータとしての自動化率が高くなるようにシステム設計することができる。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持つ。

(1)インフォメーション・ジェネレータ

調査した範囲の 4GLには、この分類に属するものは見当たらなかった。

(2)アプリケーション・ジェネレータ

①ソースコード・ジェネレータ型

データ・ディクショナリ／ディレクトリ・システム(DD/DS)を拠所にしたソースコード・ジェネレータと呼ばれる 4GLがこの分類に属する。 DD/DSとは、データが定義されたことの登録、参照（どのプログラムがどのデータを使用しているかなどの問い合わせ回答）、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用の管理機能をDDと呼び、マシンに対しての管理機能をDSと呼んでいる。 ISOでは、DD/DSのことを情報資源辞書システム(IRDS: Information Resource Dictionary System)と呼んで、標準化のための作業が最近活発に行われている。

TELONやCANO-AIDがこの分類に属する。

②オブジェクト・モジュールの動的結合型

プログラム部品をオブジェクト・モジュールの形で用意し、ユーザの指定に応じて必要なモジュールを動的に結合するものである。このような結合を可能にす

るためには、予めオプションの選択肢の数だけモジュールを用意しておき、動的に結合すべきモジュールの名称をリンケージ・テーブルの形で管理すればよい。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、オブジェクト・モジュール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、オプションの選択肢の数だけアセンブラでモジュール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。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。PRO-IVは、内部ファイルとしてVSAMのKSDSを使用しており、この分類に属する。

(3)統合化4GL

既存のDBMSユーザを対象に 統合化4GLを構築するものである。この分類に属する 4GLは、 統合化4GLと言っても、主体とするアプリケーション・ジェネレータ機能に、インフォメーション・ジェネレータとして一部の機能（検索してレポートを作成する機能）を追加した程度のものである。 CSPなどがこの分類に属する。CSPでは、 MSL(Member Specification Library)と呼ばれる開発者用ライブラリと AFL(Application Load Library)と呼ばれる実行用ライブラリを拠所にしている。

4. プロトタイピング技術の分類

現在知られているプロトタイピング技術（= プロトタイプの作成技術）は、その実現方式から次の3つに分類できる。

(1)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。従って、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。

ここで、或る言語で記述されたものが、コンパイルするだけで実行可能であるということは、記述結果に対して、マシン上で実行可能となるような解釈（= 計算モデル）が存在する、言い替えれば、これらによる記述が操作的な意味を持つということである。

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することができる。即ち、その言語で記述されたものが、主として操作的な意味を持つものがプログラミング言語であり、宣言的な意味を持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持つ言語だと言うことができる。Balzerらによれば、実行可能な仕様記述によるソフトウェア開発は図3のように表現できる[2)]。

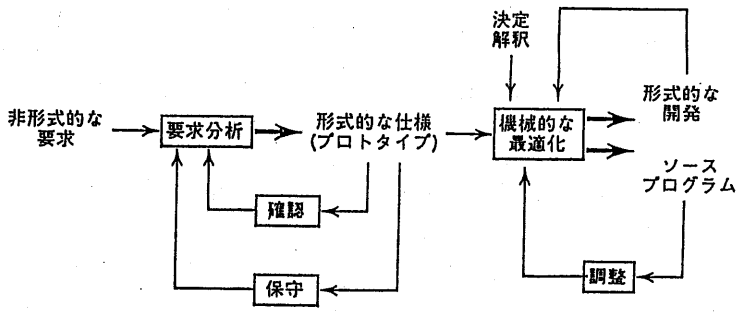


図3 実行可能な仕様記述によるソフトウェア開発

(2)プログラム言語へ自動変換される仕様記述

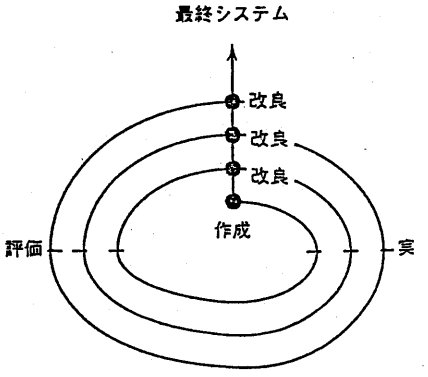
記述された要求仕様がプログラム言語へと自動変換され、得られた結果をコンパイルすれば、プログラムとしてそのまま実行可能となるシステム（= ツール）がこの範疇に入る。自動プログラミング・システムと言われるものの多くがこの形態を採っており、一般に、対象とするプログラムの範囲が狭いという欠点を持っている。従って、これを広げることができればプロトタイピング技術として利用できる。なお、自動プログラミング・システムについては、前プロジェクトで研究され、文献[8]に詳しい解説がある。

(3)機能と入出力仕様のみを確認するもの

事務処理プログラムでは、ユーザの要求仕様を確認するのに、作成するプログラムの機能とそこで使われ

る画面や帳票などの入出力仕様のみを確定するだけで、対象とするプログラムの仕様をほぼ決定できることが知られている。従って、プログラムの開発に先立って、画面や帳票などの形式のみを確認することが少なくない。これはプロトタイピングと同様の効果を狙ったものであるが、ここで使われているのは静的なモデルだけなので、その効果は不十分なものだと言わざるを得ない。プロトタイピングと呼ぶには、実物イメージの画面や帳票などを使って、対象業務をシミュレートするような動的なモデルが必要である。この種の機能を持った、ある程度の汎用的なツールの作成は可能と思われる。

プロトタイピングのプロセス



(注) Boehmが提唱したスパイラル・モデルとは異なる

図4 スパイラル・モデルに基づくソフトウエア開発

4GLは、プロトタイピング・ツールという立場から見ると、一見、3章の分類における(3)に該当するかのように思われる。しかし、そこで使われているのは静的なモデルだけなので、(3)に該当するとは言い難い。4GLには、ソースコードを生成するものとしないものの2つがある。4GLを仕様記述言語とは言い難いが、前者が3章の分類における「プログラム言語へ変換される仕様記述」に相当し、後者が「実行可能な仕様記述」に相当する。前者の例としてはLINCⅡがあり、後者の例としてはPRO-Ⅳがある。従って、4GL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、スパイラル・モデルに基づく(＝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4参照)と見なせる。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。

また、4GLの中には、開発環境と運用環境を明確に区別するシステムがある。「環境使い分け型」または「変換型」と呼ぶべきシステムがそれである。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。CSPとCASETがこの分類に当てはまる。

5．4 プロトタイピング・ツールとしての有効性からの分析

ソフトウェア生産ツールが、プロトタイピング・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充分広く、様々な要求を仕様として表現できること。

④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持つこと。

⑤要求仕様の与え方が容易で、かつ、与え方そのものの習得が容易であること。

の5つである。

4GLは、対象とするプログラムの範囲がSASなどを除けば事務処理プログラムに限られるが、上記の条件①～⑤をすべて満足している。ソフトウェア工学的には完成度は高くないが、これによって第4世代言語がプロトタイピング・ツールとして有効であることが裏付けられた。そして、現状の4GLがプロトタイピングに特に有効である状況としては、次の3つが挙げられる。

①試行錯誤的にプログラミングする場合

②画面・帳票等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・ペインターやフォーム・ペインターの採用が必須と思われる。

③環境使い分け型により開発する場合

特に、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のが有効であろう。また、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機能も有効であろう。

6．試作するシステムとその構想

IPA の技術センターでは、上記の分析結果を踏まえて、より実用的なプロトタイピング・ツール（但し、プロトタイプ）を試作することにした。試作するシステムは、 4GLの将来像だと言ってよい。また、プロトタイピング・ツールとして見れば、図５に示すように、３章の分類における(２)と(３)を組み合わせたものとなっている。

IPA の前プロジェクトでは、事務処理プログラム用の自動プログラミング・システムPAPSを開発した。PAPSは、階層的メニューとデシジョン・テーブルにより要求仕様を与えるだけで、COBOL で記述されたプログラムを完全自動で生成するシステムである。しかも、PAPSで生成されたプログラムは品質が保証されており、ユーザがデバッグをする必要がないという特徴を持っている。そこで、PAPSをシステムの核に据えたプロトタイピング・ツールを開発する。これが実現すれば、ソフトウェア開発に次のような画期的な効果をもたらすことができる。

即ち、ソフトウェア開発が、プロトタイピング機能を追加したPAPSの利用により

◆ライフサイクルの早い段階にプロトタイピングすることにより、与えた要求仕様がユーザの意図どおりであることを確認し、

◆確認された要求仕様を満足するプログラムを完全自動で生成する。このとき、

◆生成されたプログラムの品質が保証されており、ユーザはデバッグする必要がない。

となるということである。

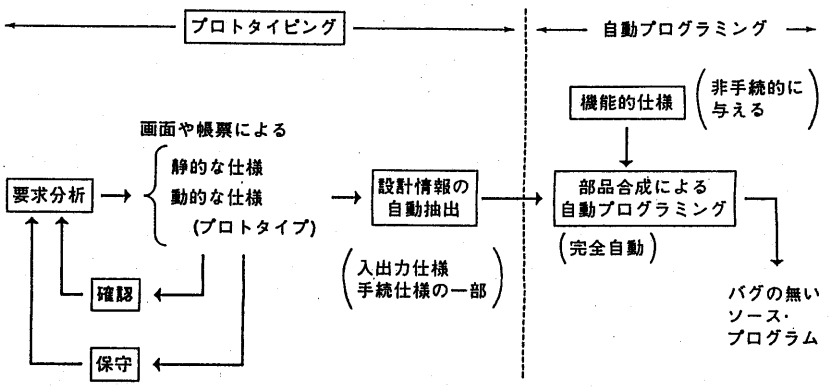


図５ PAPSによるソフトウエア開発

これを実現するために、PAPSに次のような機能を追加する予定である。

①画面・帳票などのレイアウト設計を強力に支援する機能

ディスプレイ上に画面や帳票などを実物のイメージで作成する方式（＝ スクリーン・ペインターやフォーム・ペインター）を採用する。

②画面遷移シミュレータの機能

①で得られた実物イメージの画面や帳票などを使い、ディスプレイに向かって、実際の業務と全く同じ操作によるシミュレーションを可能にする。これにより、「機能と入出力仕様の確認」によるプロトタイピングが可能となる。

③入力データのテスト表示機能

画面や帳票などに入力されたデータを表示することにより、データが正しく入力されたことを確認できるようにする。

④制約を記述する機能

プロセスやデータ間で成立すべき条件を制約として記述できれば、対象業務をモデル化することができる。このために、制約を記述する機能を設ける。

⑤リアルタイム処理プログラムをも自動プログラミングできるように機能拡張する。

現在、PAPSで自動プログラミング可能なプログラムはバッチ処理だけなので、リアルタイム処理プログラムに対しても可能となるように機能拡張する。

⑥①を知的に支援する機能

知識ベースの利用により、画面・帳票等のレイアウトに関する設計作業の負担をさらに軽減できるようにする。

さらに、プロトタイピングだけでは実現できない、次の機能を追加する予定である。

⑦性能の事前評価機能

自動生成されるプログラムの性能を事前に評価解析

する機能を追加する。

⑧与えた要求仕様の評価解析機能

与えた要求仕様そのものについての無矛盾性・完全性（曖昧さや欠落情報のないこと）・デッドロック・到達可能性などを評価解析する機能を追加する。

7．おわりに

４ＧＬを調査し、その実現方式から分析し評価した。また、プロトタイピング技術を分類し、４ＧＬの利用が分類のどれに該当するかを明らかにするとともに、４ＧＬのプロトタイピング・ツールとしての有効性を分析し評価した。詳しい内容については文献[９]または[10]を参照されたい。最後に、これらの分析と評価を踏まえて、試作すべき実用的なプロトタイピング・ツールの構想を示した。それはまた、４ＧＬの将来像であると言ってよい。

[参考文献]

[１]「第四世代言語（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).

[２]池田秀人：第４世代言語の現状と将来，情報処理学会「アドバンストデータベースシステム」シンポジウム論文集，pp.53-62，(Dec. 1988).

[３]石井義興：近代化への挑戦 なぜ第４世代言語か，Computer Report 1985年10月臨時増刊号，pp.26-32.

[４]月刊コンピュトピア編：「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).

[５]「広がるソフトウェア革命」，日経コンピュータ別冊，(10 Sept. 1986).

[６]角井正昭ほか：ソフトウェア開発変換システム，事務管理 第25巻第11号，pp.103-159(1986).

[７]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).

[８]古宮：プログラム・シンセシス法の分析と実用化への方向付け，技術センター第４回発表会論文集，情報処理振興事業協会，pp.71-85(1985).

[９]古宮：プロトタイピング技術と４ＧＬの分析に基づく試作システムの構想，ソフトウェア・ツールシンポジウム論文集，情報サービス産業協会と協同システム開発株式会社，pp.37-46(Jan. 1989).

[10]古宮：プロトタイピング技術の分析と実用化への方向付け，技術センター第７回発表会論文集，情報処理振興事業協会，(1988)(掲載予定).

[11]Martin, J.: "Application Development Without Programmers", PRENTICE-HALL, (1982).

[12]Martin, J.: "Fourth-Generation Languages", Vol.1,2, and 3, PRENTICE-HALL, (1985).

[13]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，NS2-011-001～110，(June 1986).

[14]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），NS2-101-201～292，(Sept. 1988).

[15]日経データプロ・ソフト：第４世代言語の機能モデル，NS2-106-001～009，(April 1984).

[16]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，NS2-107-001～011，(Sept. 1988).

[17]日経データプロ・ソフト：レビュー，NS2-151-151～505，(July 1987).

[18]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.

[19]「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).

[20]末舛史郎ほか：「第四世代言語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)

[21]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 第26巻第3号，pp.15-105 (1987).

[22]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.

[23]山崎明：第４世代言語（４ＧＬ）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回発表会論文集，情報処理振興事業協会，pp.179-185(1988).

[24]Yankee Group: "Exploting Software Technologies", (Dec. 1986).