ソフトウェア工学 135－5
（2001．11．21）

オンライン業務プログラムの環境独立処理方式

今城哲二　日立製作所 ソフトウェア事業部 †
／奈良先端科学技術大学院大学 情報科学研究科
吉野松樹　日立製作所 ソフトウェア事業部
大坪稔房　日立製作所 ビジネスソリューション事業部
原田晃　日立製作所 システムソリューショングループ
大野治　日立製作所 システムソリューショングループ
植村俊亮　奈良先端科学技術大学院大学 情報科学研究科

† 〒244-8555 横浜市戸塚区戸塚町5030番地 Tel：045-881-7161 E-mail：imajotji@itg.hitachi.co.jp

あらまし 10年ほど前に，第4世代言語（4GL）はメインフレームの業務アプリケーションを効率よくを構築する手段として注目を浴びていた．我々の開発した4GLは，メインフレームのオンライン環境と対話環境の両方で動作可能なアプリケーションを作るため，環境独立なオンライン業務プログラム処理方式を採用した．現在,4GLはアプリケーション開発の主流ではないが，そのテクノロジは現代のオープン環境にも有効に適用できる．
キーワード 第4世代言語，環境独立処理方式，オンライン業務プログラム，日本語プログラム

Environment-independent Online Application Program Pattern

Tetsuji Imajo　Hitachi, Ltd. Software Division †
and Graduate School of Information Science,
Nara Institute of Science and Technology
Matsuki Yoshino　Hitachi, Ltd. Software Division
Toshifusa Ootsubo　Hitachi, Ltd. Business Solution Systems Division
Akira Harda　Hitachi, Ltd. System Solution Group
Osamu Ohno　Hitachi, Ltd. System Solution Group
Shunsuke Uemura　Graduate School of Information Science,
Nara Institute of Science and Technology

† 5030, Totsuka-cho, totsuka-ku, Yokohama, 244-8555 Japan Tel: +81-45-881-7161

Abstract: About 10 years ago, the 4th generation languages (4GL) were used as an efficient way to develop online application programs. Our 4GL supported environment-independent online application program pattern to be used both online environments and interactive environments on mainframes. Now 4GL is not a main stream of application development, but the idea of environment-independent pattern is still powerful on today's open environments.
Key words: the 4th generation language, environment-independent program pattern, online application program, Japanese programming language

1. まえがき

企業や官公庁の基幹業務（経理，人事，生産管理，販売，銀行取引など）のシステム開発では，COBOLやPL/Iの第3世代言語を使い，EAGLE [1] [2]などのシステム開発支援ツール（CASEツール）を用いて生産性向上を図ってきた．この開発方式の対象業務の主要稼働環境は大型メインフレームであったが，オープン環境に対しても有効である．これについては我々の一連の論文で報告した [3] [4] [5].

別の開発方式として，プログラム言語の水準を上げるやり方がある．この言語は，簡易言語とかエンドユーザ言語[6]といわれる分野であり，1990年代前後には第4世代言語[7]という名前で呼ばれ，各ベンダが競って製品化した．この言語の対象業務は企業の基幹業務であるが，稼動環境は中小型メインフレームやオフ

ィスコンピュータが中心で，大企業の部門や中小企業の情報システム部門と，フルターンキーのディーラーが主な利用者だった．この稼動環境はオープン化の波に流され，現在は市場の表舞台では主役ではなくなった．しかし，その設計思想は貴重であり，現在のオープン環境にも適用可能な普遍的なものも多い．本論文では，その1つとして“環境独立なオンライン業務プログラム処理方式”を中心に紹介する．

この論文の構成は次のとおりである．**2.**では，1988年に出荷された中小型メインフレーム向けの第4世代言語EAGLE/4GL [8] の開発背景，特長および機能を述べる．**3.**では，1990年出荷のEAGLE/4GL後継製品 [9][10]において，その実行環境を大型メインフレーム，オフィスコンピュータ，UNIXに広げ，開発環境をUNIXとしたことにより必然的に必要となった拡張項目を論じる．**4.**では，そこで採用した“環境独立オンライン業務プログラム処理方式”について述べる．また，この方式がオープン環境でも適用可能なことを示す．**5.**では，この方式の今後の課題を明らかにし，**6.**では，本論文のまとめを述べる．

2. 第4世代言語EAGLE/4GL

2.1 開発の背景

一般的に，第4世代言語は次のように定義されている．

(1) ユーザフレンドリである．
(2) 専門家でないプログラマでも結果が得られる．
(3) COBOL（第3世代言語）の1/10の命令数で結果が得られる．
(4) データベースシステムを直接利用できる．
(5) プロトタイプを手早く作成し，変更することができる．
(6) 一般的ユーザーが2日間の学習で一応使えるようになる．
(7) COBOLの1/10以下の期間で結果が得られる，など．

現実に市販された第4世代言語がこれらの項目にすべて合致していることは少なく，いずれかに重点を置いている．

第4世代言語を大別すると，非定形的な業務を利用部門で直接作成するためのエンドユーザ言語と，コンピュータの専門家（情報処理部門）が基幹業務を開発するための高生産性言語と呼ばれるものの2種類に分かれる．

第4世代言語EAGLE/4GL（以下では4GLと略す）は後者の“基幹業務開発用の高生産性言語”を目的としている．この理由は，この言語の最初の実行・開発環境が中小型メインフレームであったことが大きい．中小型メインフレームでも比較的大きな機種のユーザにおいては，そこの基幹業務は数人～10数人の情報処理部門で開発されており，基幹業務はそれなりに難しいためエンドユーザに任せることもできず，多くのバックログをかかえていた．ここでは，一にも二にもシステム開発の抜本的な生産性向上が緊急課題であった．一方，比較的小さな機種のユーザにおいては，専門の情報部門があっても1～2人など小人数のため，自ら基幹業務を開発する余力がなく，ディーラーなど外注会社に全面的に開発を委託することが多かった．この外注金額もそう大きくはなく，競争も激しいので，外注会社にとってシステム生産性向上が最大命題だった．この2つの要求に応える言語として，4GLが開発された．

4GLの開発時に，マーケティング部門が開発部門に要求した生産性向上の目標値は，COBOLの3～4倍だった．この数値の根拠は，4GLが稼動する中小型メインフレームの新機種の販売目標に対し，それを主に販売するディーラーのフルターンキーに従事するSEの数が少なく，販売目標台数を達成するには当時のSEの生産性と比べ3～4倍の向上は必要という経営上の要求による．この値は先に示した第4世代言語の定義には至っていないが，他社協業上“第4世代言語”という分野の製品も必須だったため，あえて製品名に“第4世代言語（4GL）”の名前をつけ，10倍という数値は努力目標とした．4GLの前にEAGLEをつけたのは，日立の開発支援ツールの総称がEAGLEであったことによる．

2.2 特長と機能

4GLの特長は次の6つである．

(1) 対話処理によるシステム開発

4GLの開発時の動作環境は，日立の中小型メインフレームのOSであるVOSK[11]の対話環境である．VOSKの対話環境は，大型OSのTSS環境と類似しているが，OS本体にデータベース機能を持たせたことにより，TSSとは別環境だったオンライン業務環境をも包含している．この対話環境で日本語メニューの開発環境画面（図1）により，システ設計から運用・保守

までの一連の作業が可能である．海外用には英語メニューも用意している．

(2) プロトタイピングによる画面・帳票定義

図2に示すように，業務画面・帳票イメージをフルスクリーン画面上でラフスケッチ定義することにより，直ちに確認画面を表示するプロトタイピング技法を採用している．これにより，システム設計の段階で最終的な画面・帳票様式，大きさ，色などの詳細な仕様確認が可能であり，早期に入出力フォーマットが確定できる．

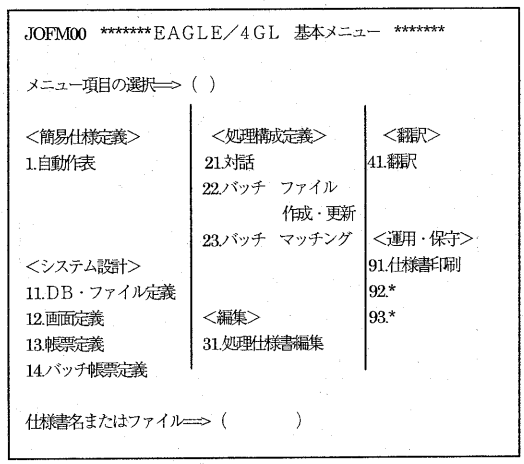

図1 EAGLE/4GLによる業務選択メニュー 目的業務の番号を選択することで業務を開始できる．

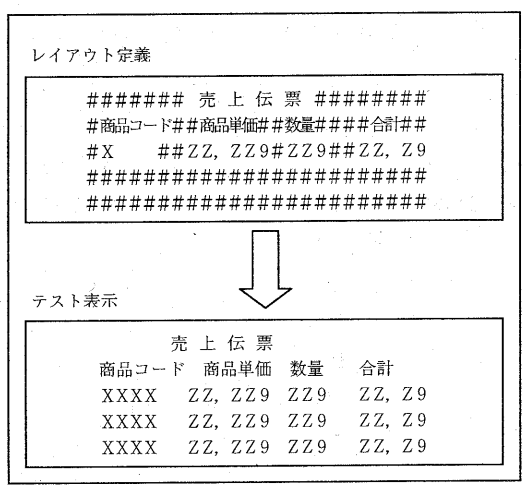

図2 プロトタイピングによる画面・帳票定義 画面・帳票の入出力項目を指定するだけで，それらのイメージをその場で確認できる．

(3) 処理のパターン化

4GLでは，大型メインフレームのシステム開発支援ツールEAGLEのバッチ22パターンとオンライン5パターンを参考にし，図3に示すようにコンピュータ業務処理を3つのパターンに集約した．

(a) 対話パターン

(b) ファイル作成・更新パターン（バッチ）

(c) マッチングパターン（バッチ）

このパターンと一致する業務処理については，基本アルゴリズムを考えることなく，処理構成メニューの空きを埋めていくだけでプログラムが作成できる．各パターンで可能な処理の種類は以下の通りである．

(a) 対話パターン

①業務振り分け，②問い合わせ，③ファイル更新，

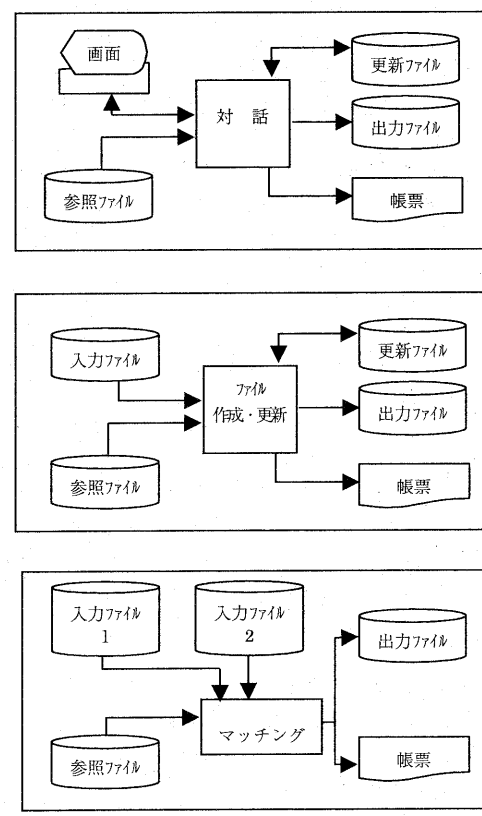

図3 EAGLE／4GLの処理パターン 対話1パターン，バッチ2パターンでプログラムが組める．

④ファイル更新と帳票出力，⑤データエントリ

(b) ファイル作成・更新パターン（バッチ）

①ファイルの分配と抽出，②ファイルの集約と照合，③ファイルの更新とレコード追加，④帳票出力，⑤「ファイルを更新しながら帳票出力する」などの複合形

(c) マッチングパターン（バッチ）

①入力ファイル2本の照合（1：N），②ファイル照合時のエラーリストの出力処理

(4) 日本語プログラミング

仕様書を示す．図4の上部のメニューは処理構成定義で，3つのパターンごとに異なる図柄で表示され，そこにはプログラムの入出力（ファイル，画面，帳票など）の仕様書名を投入する．この処理構成定義から図4の下部に示す4GLの処理仕様書（ソースプログラム）が自動生成される．4GLを使用する設計者（プログラマ）は，この自動生成された仕様書を基にエディタを用いて業務仕様を追加する．この追加のために，各種の記述文や関数を用意している．これらの処理記述文(転記,四則演算,条件により2分岐(if文に相当)，条件により多分岐（case文に相当），ループ（while文やfor文に相当）など）と関数名は日本語である．画面・帳票・DB・ファイル・データの項目名称にも日本語が書けるので，英語をベースとした第3世代言語とくらべ，日本語を使う者にとって親和性が強く読みやすく書きやすい．なお，日本語以外に英語を基調とした構文も用意している．

(5) 簡易仕様自動定義（自動作表）機能による業務仕様自動生成

帳票を出力するプログラムの開発に有効な機能であり，入出力構成，帳票の形，および項目情報を記述するだけで，帳票出力プログラムを完全自動生成可能とした．すなわち，プログラミングレスの実現である．これを適用できるのは，バッチ帳票の中でもよく使われるファイル内容を直接印字する一覧表や，小計や合計を印字するような集計表であり，図5に示す入出力構成が条件である．

(6) COBOLソースとXMAP定義文を生成

新しい言語の開発において次の4つの大きな作業が必要である．

①言語仕様の開発，②コンパイラの開発，

③実行時ライブラリの開発，④開発環境の開発

新言語普及のためには実行時の性能と品質も重要である．すなわち，②で生成するオブジェクトの効率を

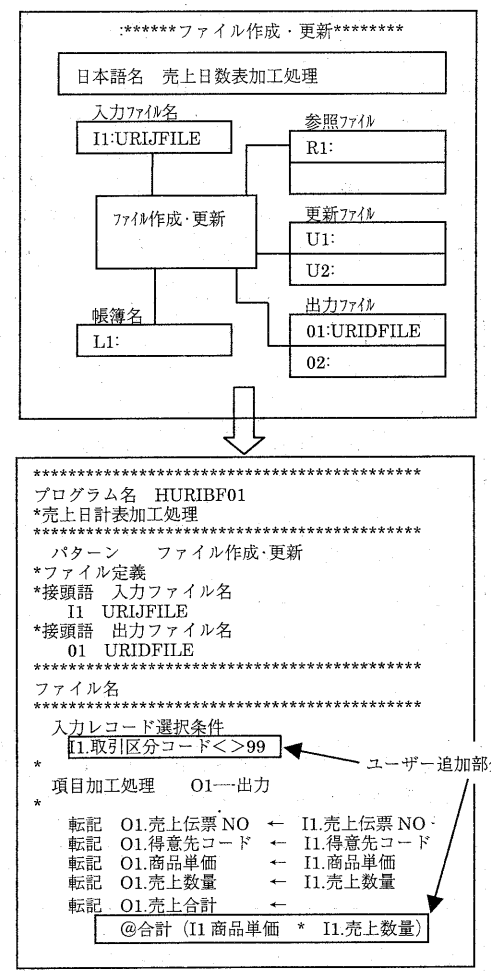

図4　処理仕様書　　プログラム入出力情報を指定し，処理仕様書を自動生成する．データ項目の名称や処理の指示（文）に日本語が使える．

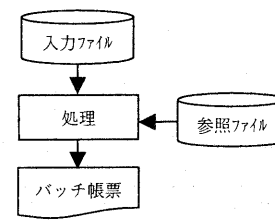

図5　簡易仕様定義（自動作表）入出構成　　この入出力構成であれば，簡易仕様定義（自動作表）による開発が行える．

良くし，③では性能と品質に特に配慮する必要がある．これらの開発負荷を低減するため，4GLではCOBOLのソースプログラムを生成し，実行時の性能と品質の課題はCOBOLに頼ることにした．このことにより，4GLの開発においては②と③の作業が大幅に低減でき，①と④に注力できた．また，COBOL生成としたことにより，新言語に対するユーザの“将来とも存続するか”という危惧に対しても，“いざとなったらCOBOLで保守可能”という逃げ道を用意できた．

また，4GLでは画面・帳票の実行時ライブラリの開発量も多大なので，4GLの画面・帳票定義から，汎用の画面・帳票製品であるXMAPの定義文を生成するようにし，作業量の低減と実行時の性能・品質の確保が図った．

2.3 生産性実績

4GLの生産性を評価するため，受注・販売管理システムを4GLとCOBOLの両方で記述実験した．

プログラムの内容は，受注データの入力や発注商品一覧表，売上日報印刷処理など9本である．ただし，4GLとCOBOLのデータ定義部分は同等であるため，手続き部（procedure division）に記述したステップ数を比較した．評価結果を表1に示す．この記述実験では，4GLの記述量はCOBOLに比べ，バッチ処理で1/10，対話処理で1/3に減少した．

表1　4GLとCOBOLの生産性　　受注・販売管理システムを4GLとCOBOLの両方で記述した結果である．

項番	パターン	手続部記述量		比率
		COBOL	4 GL	
1	バッチ	96	11	1/9
2	バッチ	11	0	—
3	バッチ	59	6	1/10
4	バッチ	113	10	1/11
5	バッチ	21	1	1/21
6	バッチ	48	9	1/5
7	対　話	232	72	1/3
8	対　話	197	72	1/3
9	対　話	116	43	1/3

3.　4GL実行環境の拡張と開発環境の一本化

3.1 多様な実行環境への適用

1990年代はじめには，第4世代言語の必要性，すなわち，この言語によるシステム生産性向上の期待は中小型メインフレームだけでなく，大型メインフレームやオフィスコンピュータ，さらにUNIXなどのオープン環境と多様な分野に広まり，そのサポートは緊急課題であった．また，SEは1つの分野の仕事だけを続けることはまれで，複数の分野のシステム開発に携わることが多い．このようなニーズから，マーケティング部門から製品開発部門への要求仕様は，“多くの分野に適用可能な1種類の第4世代言語で，開発環境も1つだけとする”ということであった．

これらの条件をすべて満たすために4GLの開発環境をUNIXに移行し，その開発環境で対象となる実行環境（大型メインフレーム，中小型メインフレーム，オフィスコンピュータ，UNIXなどのオープン環境）のCOBOLソースプログラムと画面・帳票・DB・ファイルなどの定義文を生成するようにした．

さらに，UNIX環境のCOBOLコンパイラやXMAPを用いてオブジェクトを生成し，UNIXで実行することにより，これらの単体テストを可能とした．これが，実現できたのは，COBOL，XMAP，OLTP，DBなどメインフレームと互換性を有する主要ミドルウェアが順次UNIXで稼動したことによる．

3.2 必然的に必要となった拡張項目

実行環境の拡大と開発環境のUNIX化により，必然的に必要となった主要な拡張項目を次に述べる．

3.2.1 開発環境メニューのGUI化

1980年代の終わり頃にUNIX上の標準GUI機能であるMotifの仕様が決まったので，4GLの開発環境メニューもそれを用いた（日本国内でのMotif の最初の利用者）．このため，VOSKの4GLの開発環境メニュー（CUIインターフェース）と比べユーザインターフェースが抜本的に改善された．

ただし，CUIとGUIのインターフェースが全く異なり，UNIX上の記述言語もC言語しか選択の余地がなかったので，VOSKの4GLのコーディング（メインフレームのシステム記述言語とCOBOLとで記述）は流用できず，全面的に再設計・リコーディングとなった．

3.2.2 メインフレームのオンライン環境の業務プログラムサポート

オンライン基幹業務はVOSKでは対話環境で実行可能であったが，大型メインフレームにおいては，対話環境と類似環境であるTSS環境では一部の簡易な業務だけが実行され，多くの業務はオンライン環境で実行される．4GLでは，このオンライン環境の業務プログラムをサポートする必要がある．これについては，5.で詳しく述べる．

大型メインフレームでオンライン環境が業務に使われる代表的な理由に次のものがある．

(1) 信頼性：TSS環境でDB共用や異常時の回復などすべてユーザ責任であるが，オンライン環境ではそれらの面倒はオンライン制御プログラム（DCシステムあるいはOLTPともいう）がみてくれる．

(2) 資源の有効活用と応答時間の平準化：オンライン制御プログラムは多くの端末からの大量のトランザクションをキュー管理し，限られたシステム資源を有効に利用しながら，応答時間が均一になるように適切にスケジューリングして，業務プログラムを実行する．TSS環境では，端末ごとに1つの空間を占有するのでシステム資源を大量に必要とする．

3.2.3 DB（SQL）サポート

VOSKのDBはリレーショナルライクなDBであり，ファイルのようにread/writeインターフェースでアクセスする．一方，大型メインフレームの業務にはDBが必須で，4GL開発当時はSQLが普及段階に入っていた．4GLでSQL機能を制限せずに，サポートする必要がある．このために，すべてのSQL構文に対応した日本語SQL構文を4GLでサポートした．

3.2.4 生成したソースなどの対象環境への転送

UNIX環境の4GLはメインフレームの3種類のOS（VOS3，VOS1，VOSK）およびオフィスコンピュータのOS（MIOS7）に対応したCOBOLソースプログラムや各種の定義文（パラメタ）を生成する必要がある．さらに，それらをファイル転送などで，個々のOS環境に送りコンパイルするなどのしかけをサポートする必要がある．

4.環境独立オンライン業務プログラム処理方式

4.1 対話環境とオンライン環境における業務プログラムの基本構造の相違

典型的なオンライン業務である問合せ応答処理を例にとって，対話環境とオンライン環境における業務プログラムの基本構造の相違を説明する（図6）．

(1) 対話環境

業務プログラムの起動は端末からのコマンドあるいはメニュー画面で行う．起動されたプログラムは1つの空間に常駐する．プログラムから業務画面を出力し，端末からの入力を待ち，入力されたらDB入力など何らかの処理をし，端末の画面に結果を表示する．それが可（OK）の場合はその旨の応答を受けて，必要ならばDBを更新する．結果が否（NG）の場合は，再度処理をやり直す．

この対話環境の業務プログラムの基本構造は“画面を出力し入力⇒処理⇒画面を入力し出力⇒・・・”という形態である．

(2) オンライン環境

端末から開始コマンドをオンライン制御プログラムが受けると，定められた初期画面を表示する．このあと，この画面に端末側が応答し必要な入力を行う．これをトランザクションとしてオンライン制御プログラムが受け取り，キューに入れ，スケジューリングし，空間が空いたらそのトランザクションに関係づけられた業務プログラムを起動する．業務プログラムはトランザクションをメッセージとして入力（receive）しDB入力など必要な処理をし，応答メッセージ（画面）を出力（send）し，プログラムは終了する．このプログラムが一時的に占有していた空間は，オンライン制御プログラムが別の端末対応の業務プログラムに割り当てる．出力メッセージは出力キューを経由して端末に送られる．

すなわち，オンライン環境の業務プログラムの基本構造は“画面入力⇒処理⇒画面出力”である．

問合せ応答業務では，出力した画面に端末が応答すると，それがトランザクションとなり，それに関係づけられたプログラムがスケジューリングされて起動される．そのプログラムはDB更新などの処理を行う．2つの業務プログラム間の共有情報の受け渡しは，オンライン制御プログラムが仲介する．

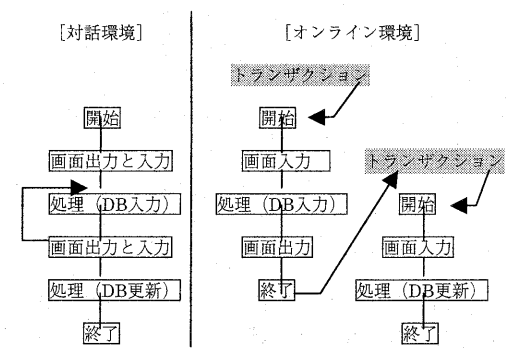

図6 業務プログラムの基本構造の相違 対話環境ではプログラムは常駐しているが，オンライン環境で常駐するのは入力から出力までの間だけである．

4.2 環境独立な処理方式

VOSKの4GLでは，バッチ2パターン，対話1パターンの3パターンだったが，UNIX版4GLでは，問合せ応答パターンを追加した．このパターンはオンライン環境と対話環境（TSS環境も含む）の両方に適用できる．この処理構造は図7に示す．

1つのオンライン業務は複数の画面で構成されることが多い．今までのオンライン業務プログラムの作り方は，業務プログラムの基本構造が“画面入力⇒処理⇒画面出力”となっていることに起因し，複数画面の制御と個々の画面処理が渾然一体で記述されていた．

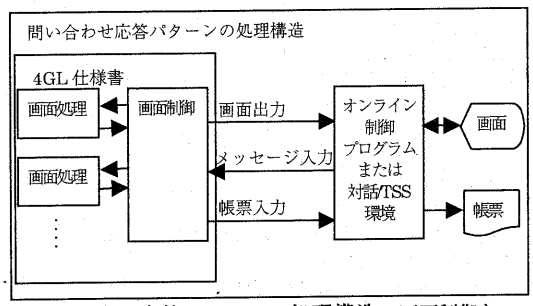

図7　問合せ応答パターンの処理構造　画面制御と個々の画面処理構成されている．

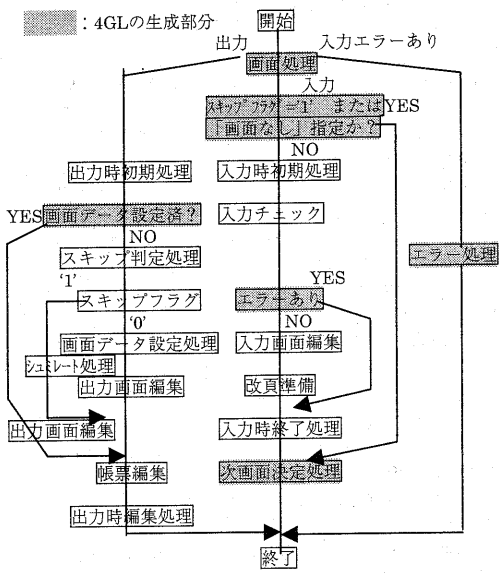

図8　問合せ応答パターンの処理概要　影付きの部分が画面制御部分であり，白地の部分が画面処理あるいは帳票処理である．

```
**************************************************
処理仕様書                              ＜処理仕様書名＞
**************************************************
* コメント　＜コメント＞
  パターン              問合せ応答
    [使用する画面，ファイル，及び帳票の指定]
    [起動するトランザクションの指定]
    [DAM ファイル，TAM テーブル，一時記憶データ
    受け渡し領域の指定]
  画面遷移定義
    [画面の遷移情報の定義]
  画面制御
    [PF キーの定義]
**************************************************
データ定義
**************************************************
  AP 間連絡領域
    [AP 間通信メッセージを受け取る領域の定義]
**************************************************
画面処理－Sn＜画面処理名＞[画面処理ｵﾌﾟｼｮﾝの指定]
**************************************************
**************************************************
データ定義
**************************************************
  通信連絡領域
    [画面処理間での連絡用作業領域の定義]
  作業領域
    [処理で使用する作業領域の定義]
  遷移条件定義
    [遷移条件名に対する遷移条件の定義]
**************************************************
手続き
**************************************************
  出力時初期処理
    [画面表示処理の最初に行う処理の記述]
  出力時終了処理
    [画面表示処理の最後に行う処理の記述]
  スキップ判定処理
    [画面の表示，スキップの判定処理の記述]
  画面データ設定処理
    [改頁構造に該当する画面処理の中で最初の画面出
    力時一度だけ行う（全ページ設定処理の記述）の記
    述]
  出力画面編集
    [表示する画面の編集処理の記述]
  シミュレート処理
    [画面を表示しない時の記述]
  帳票編集
    [帳票に出力するデータの編集処理の記述]
  入力時初期処理
    [画面データの入力直後に行う処理の記述]
  入力時終了処理
    [画面入力処理の最後に行う処理の記述]
  入力チェック
    [入力データのチェック処理の記述]
  入力データ編集
    [入力データに対する加工処理の記述]
  改頁準備
    [改頁に対する加工処理の記述]
  サブ手続き　＜サブ手続き名＞
    [画面処理－Sn での共通処理の記述]
```

図 9　問合せ応答パターンの処理仕様書　手続きの[ ]の中の処理を記述していけば処理仕様書が完成する．

4GLの問合せ応答パターンでは,①画面制御と画面処理は分離して記述し,②複数のプログラムに分離されていた同じ画面の出力処理と入力処理を 1 つにまとめて記述するようにした.

画面制御については論理的な関係と環境（オンライン制御プログラムあるいは対話・TSS環境など）を指定するようにし,環境の相違は4GLが生成する画面制御プログラムで吸収するようにした.これにより.4GLでは個々の画面に対する個別処理だめを記述すればよい.問合せ応答パターンの処理概要図を図8に示す.これに対応する4GLの仕様書を図9に示す.

4GLは画面制御と個々の画面処理を別々のCOBOLプログラムとして生成する.

4.3 環境独立オンライン業務プログラム処理方式のオープン環境への適用

UNIX環境とPC環境(Windows)にOLTP,XMAP,COBOL,DBのミドルウェアがサポートされたので,基本的な部分はほとんど変えずに 4GL の問合せ応答パターンはこれらの環境に適用可能となった.ただし,XMAP 自身は GUI をサポート済みだが,4GL の画面定義では CUI しかサポートしていないので,実際の適用はメインフレームからのストレート移行なに限られている.

5. 問題点と今後の課題

5.1 問題点

UNIX 版 4GL の開発を決めた 1990 年頃は,4GL が稼動するために充分なメモリ量があり,GUI 機能を備えたオープン環境 OS は UNIX しかなかった.しかし,その後の Windows の進歩と搭載 PC の能力向上により,開発環境は PC が当たり前になり,UNIX 版 4GL は普及しなかった.PC への移植も検討したが,GUI 機能の相違が大きく開発量が膨大で,それはあきらめた.Windows と PC の進歩の予測誤りと UNIX 普及への過度の期待が敗因である.

5.2 今後の課題

画面制御と画面定義を分離した環境独立処理方式は,もともとCOBOL用にオンライン環境とTSS環境に適用した[3]のが始まりで,4GLだけでなく他の言語にも適用可能である.この方式はJavaでも実験済みで,オープン環境でも適用可能なことが分かるっている[12] [13].現在はEJB環境での適用を検討中であり,EJB向用支援ツールの開発が課題である.

6. むすび

本論文では,第4世代言語EAGLE/4GLについて述べ,そこで開発された複数OSのオンライン環境と対話（TSS）環境に独立なオンライン業務プログラム処理方式について論じた.この方式は他の言語やオープン環境にも適用できる.現在はEJB環境での適用を検討中で,支援ツールのサポートが課題である.

参考文献

[1] 葉木洋一,今城哲二,津田道夫,仁平博三：システム開発支援ソフトウェア“EAGLE”,日立評論,第 66 巻 3 号,pp.19-24,1984 年.3 月.

[2] 葉木洋一,他：システム開発支援ソフトウェア“EAGLE”－EAGLE 拡張版 EAGLE2,日立評論,第 68 巻 5 号,pp.373-378,1986 年.5 月.

[3] 大野治,降旗由香里：ソフトウェア標準化のためのプログラムパターンの作成,情報処理学会研究報告 98-SE-121,pp.171-178,1998.

[4] 大野治,降旗由香里,小室彦三,今城哲二,古宮誠一：多次元部品方式によるソフトウェア開発の自動化――バッチプログラム用スケルトンの作成とその充分性――,電子情報通信学会論文誌,第 J83-D-1 巻,第 10 号,pp.1055-1069,2000 年 10 月.

[5] 大野治,小室彦三,降旗由香里,渡部淳一,今城哲二：多次元部品方式によるソフトウェア開発の自動化――自動生成系の開発と評価――,電子情報通信学会論文誌,第 J84-D-1 巻,第 9 号,pp.1372-1386,2001 年 9 月.

[6] 酒井紘昭（編著）：簡易言語とエンドユーザズ言語,昭晃堂,p.167,1984.

[7] Martin, J.：The 4th Generation Language.

[8] 今城哲二,秋山美登,北尾修治,里本健,脇坂隆則,大西俊治：VOSK 第 4 世代言語“EAGLE/4GL”,日立評論,第 71 巻 11 号,pp1119-1124,1989 年 11 月.

[9] 西尾高典,吉野松樹,他：アプリケーションの分散開発を実現する第 4 世代言語,日立評論,第 75 巻 9 号,pp.605-610,1993 年 9 月.

[10] 吉野松樹,田村和敏,稲益良夫：ソフトウェア開発支援ツール“SEWB3,EAGLE/4GL”の機能と特長,日立評論,第 75 巻 11 号,pp.727-734,1993 年 11 月.

[11] 矢嶋廣,長谷川栄,風間順一,山口康隆,高崎繁夫,加藤礼吉：オペレーティングシステム VOSK の基本制御方式,日立評論,第 71 巻 11 号,pp1111-1118,1989 年 11 月.

[12] 鈴木文音,大坪稔房,勝瑞雅,湯裏克彦,津田道夫,宮崎肇之,降旗由香里,小室彦三,大野治：Java によるシステム開発設計と COBOL による現行系との比較,情報処理学会オブジェクト指向シンポジウム 99,1999 年 7 月.

[13] 藤原晃,楠本真二,井上克郎,大坪稔房,湯裏克彦：複雑度と機能量に基づくアプリケーションフレームワークの実験的評価,オブジェト指向最前線 2001 情報処理学会 OO2001 シンポジウム,pp.85-90,近代科学社,2001.