第3編

ソフトウェア展望

「完全になったもの，熟したものは，みな一死ぬことを願う」
そうおまえは語る。だから葡萄を摘む鋏はしあわせだ。
それに反して，成熟に達しないものはみな，生きようとする。
いたましいことだ。

ニーチェ
「ツァラトゥストラ」
（手塚富雄訳）

第11章

エンドユーザコンピューティング

11.1 構文的パラダイム

★

業務の言葉や業務のメタファを用いて表現された
プログラムは，わかりやすい。

11.1.1 エンドユーザコンピューティングの動向

すでに第1編で述べたように，エンドユーザコンピューティングの傾向が強まってきた要因として，次のようなものがある。

① ハードウェアの低廉化と共に，ブック型コンピュータ，パソコン，WSなどの普及がめざましく，OA分野を中心にエンドユーザが急増している。「物のパーソナル化」が「情報のパーソナル化」を促進している。

② コンピュータの利用形態が，大型機によるホスト集中型からパソコン，WS主体のネットワーク型に移行し，分散コンピューティングの促進により，エンドユーザがコンピュータを直接操作する機会が急増している。「システムのグローバル化」が「情報のグローバル化」を促進している。

③ コンピュータの利用目的が，従来の業務の合理化からSISやCIMなどの経営戦略の実現に広がるにつれて，非定形業務が増加し，さまざまな業務担当者がエンドユーザとしてコンピュータを利用しはじめている。

11.1.2 エンドユーザコンピューティングのためのソフトウェア開発

オフィスにおけるエンドユーザの業務を大ざっぱに分類すると，従来は定形業務と非定形業務に分けられた。定形業務はビジネス分野の帳票処理に代表されるような企業の基幹業務にかかわるものである。このようなプログラムは，通常は，エンドユーザの要求をききながら，企業内の情報システム部門が開発あるいは導入して，エンドユーザに提供する。保守，拡張も情報システム部門が行う。エンドユーザは定められたユーザインタフェースに従ってコンピュータを使用するだけであった。一方，非定形業務については，表計算に代表されるような市販のOAソフトウェアを購入して利用してきた。

しかしながら，コンピュータの利用方法が，業務の効率化から戦略的情報システムへと発展していくにつれて，基幹業務と個人の非定形業務の連携が必要になってきている。この傾向は，コンピュータシステムが分散コンピューティングの方向に進展するにつれていっそう加速されている。その結果，従来の表計算ソフトウェアと基幹業務データベースをつないだり，データベース検索ソフトウェアにデータの加工，編集機能を追加したりして，意思決定支援システムを作り上げることが重要になっている。さらに，CSCWのように，別々の場所にいるグループが協力しあって意思決定をしていくためのグループウェアが求められている。

このような状況下でタイムリーに必要なソフトウェアを手にいれていくためには，もはや情報システム部門が提供するシステムや市販のソフトウェアパッケージを単に利用するだけでなく，業務担当者は，積極的に自ら使うソフトウェアの開発に関与していく必要がある。第3章でソフトウェア危機回避のシナリオの1つにエンドユーザコンピューティングを挙げたのはこのためである。

11.1.3 エンドユーザ指向言語

このようなトレンドに対応した言語について分析するために，次のような3種類の図式で考える。いずれも右側がエンドユーザコンピューティングを指向した言語である。

- ●手続き型言語　vs.　非手続き型言語
- ●コンピュータ言語　vs.　対象世界記述言語

●システム記述言語　vs.　応用記述言語

この各々についての考察を以下に述べる。

(1)　手続き型言語　vs.　非手続き型言語

これはパラダイムによる分類である。現在広く普及している手続き型のパラダイムは，対象とする問題の記述には適切でないことが多い。この隔たりはセマンティックギャップといわれるが，このギャップを埋めるためにプログラムを非手続き的に記述する新しいパラダイムの研究が行われ，宣言型の言語が開発されてきた。

(2)　コンピュータ言語　vs.　対象世界記述言語

これは言語の役割による分類である。言語は本来的にコミュニケーションの道具であるが，コンピュータ言語の場合は，コミュニケーションの相手がコンピュータであるため，これまで従来のコンピュータアーキテクチャと相性の良い手続き型言語が用いられてきた。これに対し，対象世界記述言語は，コミュニケーションの相手が人であるため，対象世界を抽象的に表現するモデリング技術とそのモデルに基づき問題を厳密に記述する仕様記述言語として発展してきた。最近，オブジェクト指向概念が，両者の橋渡しとなりうる単純かつ強力な計算モデルとして注目されている。

(3)　システム記述言語　vs.　応用記述言語

これは，言語の用途，すなわち記述対象による分類である。システム記述言語は，OS，データベースなどの基本ソフトウェアの開発に使用するため，実行効率が重視され，アセンブラやコンピュータアーキテクチャと相性の良い手続き型言語が用いられる。これに対し，応用記述言語はアプリケーション対応に適した言語が用いられる。大規模高信頼ソフトウェアには実行効率重視のため COBOL や PL/I が使用されることが多いが，一般には，開発，保守容易性が重視され，目的に応じた特定分野/業務向きの簡易言語として，4GL，エンドユーザ言語，DB 検索言語，表言語，ルールベース言語などが用いられはじめている。

11.1.4 エンドユーザコンピューティング技術

エンドユーザが業務のコンピュータ化を行えるための現実的な技術として，次のようなものがある。

① 日本語プログラミング
② ビジュアルプログラミング
③ 第4世代言語
④ 人工知能技術

このうち，最後の人工知能についてはすでに第9章で述べた。①，②，③は，いずれも脱プログラミングのコンセプトに基づいているが，実際には「脱プログラミング言語」のレベルであり，プログラミングの概念やセンスが不要というわけではない。見かけ上の使い勝手を向上させているという意味において本書ではこれらの技術を「構文的パラダイム」と呼ぶ。自ら作って使うという意味では，エンドユーザコンピューティングというよりもユーザオウンコンピューティングといった方が良い。

ユーザが作らないで使うだけのOA用標準パッケージ等はここでは述べないが，ハイパーテキストや表計算，データベース検索，意思決定支援などの分野もユーザオウンコンピューティングの観点で注目していく必要がある。

11.2 日本語プログラミング

★ *日本語で表現されたプログラムは，わかりやすい。*

職業的プログラマでないエンドユーザにとって，プログラムの日本語表現は書き易さ，読み易さの点で魅力的である。特にアプリケーションプログラムの保守性に不可欠であるプログラムの理解容易性の向上には，プログラム構造がきれいであることと共に，プログラムの各文の意味がわかりやすいことが重要である。

プログラムの日本語表現には次のようなレベルがある。

①既存のプログラミング言語でデータ名や手続き名に日本語が使える。
②既存のプログラミング言語の if, end などのキーワードも含め，すべて日本語で記述する。
③既存のプログラミング言語相当の記述レベルで独自の文法規則を持つ日本語プログラミング言語で記述する。
④仕様記述言語として日本語を導入し，プログラムを自動生成する。

現在の実用レベルは，②，③が主で，4GL に組み込まれているものが多い。今後は④が主流になっていくと思われるが，次のような技術課題を克服する必要がある。

①業務用語をそのまま使用できるようにするための，業務用語辞書の作成保守機能
②日本語記述レベルで閉じたシステムとし，それから変換されるプログラミング言語の知識を不要とするための，開発保守環境
③日本語表現が持つあいまい性を避け，かつ自然な表現ができる文法規則

日本語化が進んでいる分野としてデータベース検索のコマンドインタフェースがある。たとえば，データベース検索用自然語インタフェース HNLDB（日立）では，

「箱根の宿泊先を費用の安い順にだせ」

という問い合わせ文を入力すると，システムが

```
select 宿泊先テーブル．宿泊先 from 宿泊先テーブル
  where 宿泊先テーブル．所在地＝'箱根'
    order by 宿泊先テーブル．費用 desc
```

という SQL 文に変換して実行してくれる。本システムでは，上記の課題を克服するために

①入力例文集からの問い合わせ文の選択と修正機能，
②システムの解析結果（検索条件，対象）の確認と修正機能，

③あいまい語，同義語の登録機能，

などをユーザに提供して，日本語インタフェースを使いやすくしている。

日本語プログラミングのニーズが大きいもう1つの分野は，第3次銀行オンラインシステムのようなメガステップオーダの大規模高信頼ソフトウェアの開発である。これまでは，ユーザの情報システム部門，ソフトウェアハウス，メーカのSE部門が協力して，COBOLやPL/Iなどのプログラミング言語を用いて開発してきたが，これまで以上の大規模なシステムは，以下のような理由で従来方式での開発は無理である。

①計画段階から稼動までの期間が長く，数年先を見た計画は不可能。
②開発管理が規模的に不可能。
③開発中の機能変更や稼動後の機能拡張への迅速な対応が不可能。

これに対し，日本語プログラミング技術を適用すれば，プログラムの仕様が業務仕様書レベルになり，エンドユーザがプログラムを開発したり，保守できるので，上記の問題は軽くなる。その1つの試みとして，筆者等が開発した業務仕様記述言語システムSPECTRUMを紹介する。本システムでは，文型を用いた限定日本語の仕様記述言語を開発し，メニュー選択方式で仕様記述を誘導する方式を取っている。画面の例を**図11.1**に示す。画面の左上と左下のウィンドウが日本語仕様の記述例である。本システムは以下の特徴がある。

①日本語文型はシステム組込みの21種類の組み合わせで充分記述可能であるが，ユーザ定義も可能。(画面の上中央に文型例)
②限定日本語文の目的語をオブジェクト，動詞をメソッドとみなし，表形式のオブジェクト指向プログラミング言語に内部変換することにより，意味を厳密に規定。(画面の右上に例)
③この表形式のオブジェクト指向プログラムを業務用語辞書やプログラム部品ライブラリを用いてPL／Iプログラムに変換。(画面の右下に例)

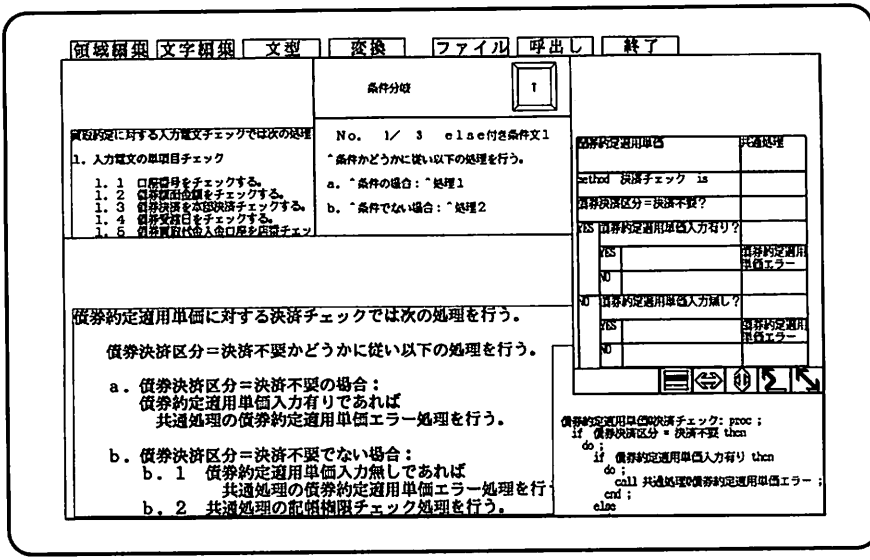


図11.1　日本語仕様記述言語システムSPECTRUMの画面例

11.3 ビジュアルプログラミング

> ★ *図表で表現されたプログラムは，わかりやすい。*

従来のプログラミング言語の難しさを回避する方法として，日本語プログラミングでは記述の形式性を緩和することによって記述容易性と理解容易性を実現しているが，ビジュアルプログラミングでは記述の線形性の緩和によってそれを実現する。最近のビットマップディスプレイ付きワークステーションの普及と共に，以下の3つの観点からのプログラムの視覚化の研究が活発に行われてきた。

①視覚的ユーザインタフェース
②視覚的編集
③視覚的言語

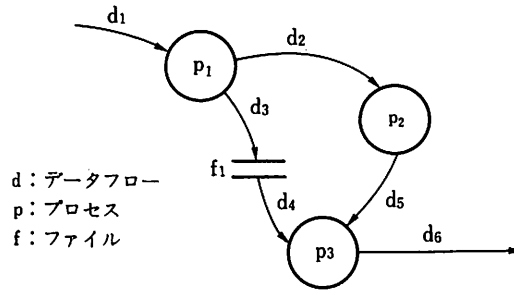
第一の視覚的ユーザインタフェースは，プログラミング環境を用いてプログラムを開発するときに，システム側から提供される種々の情報をグラフィカルに表示し，ユーザ側からの入力もそのグラフィカルな画面上で行えるようにするものである。代表的なシステムとして Smalltalk-80 などがあるが，最近では一般化している。

第二の視覚的編集は，本来テキスト表現を基本に作られたプログラミング言語を用いてプログラムを作成する場合に使用するテキストエディタに代わるものである。テキストエディタでは，プログラムを単なる文字列として扱うため，文法的エラーを作り込みやすい。そこで，プログラミング言語の文法規則をシステム側が知識として保有しておき，if 文などの基本構造をテンプレートとして表示し，それに穴埋めさせる方式で文法的に正しいプログラムを誘導する構造エディタが普及し始めている。筆者らもこのテンプレート方式とテキスト入力方式を融合した構造エディタ PARSE (Production And Reduction Structure Editor) とその構造エディタを種々のプログラミング言語対応に自動生成する構造エディタジェネレータ PARSE-G を開発した経験がある。さらに，テンプレートとして，5.2 節で述べた PAD などの構造化図式を用いた図式エディタも利用されている。

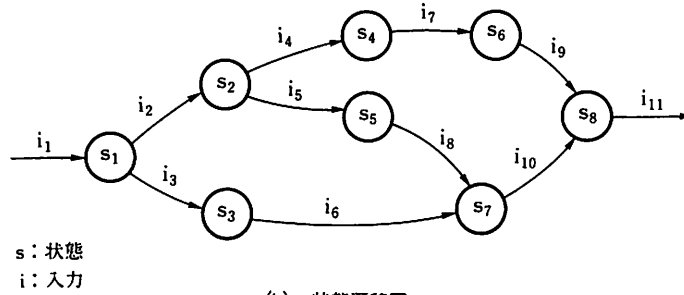
第三の視覚的言語は，本来的に視覚的表現を基本とした言語であり，その視覚化に用いる図式の表現形式の違いにより，フォーム指向言語，グラフ指向言語，アイコン指向言語に分類される。

フォーム指向言語は，表形式を基本とするもので，OA 分野のスプレッドシートを用いたフォームベースプログラミングが代表的である。先に述べた SPECTRUM システムの表形式オブジェクト指向言語もその一例である。このシステムでは限定日本語による仕様記述のほかに，図 11.1 の右上のウィンドウに示すような表形式で直接仕様を記述するためのエディタを用意している。

グラフ指向言語は，ノードとアークで構成される有向グラフ表現を基本とするもので，**図 11.2** に示すデータフロー図，状態遷移図などがある。実行可能な仕様記述言語の例として，Statechart がある。これは，わかりやすさの利点を持つ視覚性と厳密さの利点を持つ形式性をあわせ持つ視覚的形式性という概念に基づき，状態遷移図を拡張したものである。**図 11.3** にディジタル腕時計の 4 種類のボタンを押したときの表示機能を Statechart で記述した例を示す。主な



(a)　データフロー図



(b)　状態遷移図

図11.2　グラフ指向言語に用いられる基本表現の例

特徴は，状態遷移図の入れ子構造（ネスト構造）による分割統治手法の導入，直交，直積関係の表現による非同期処理の導入などである。

アイコン指向言語は，ビジュアルオブジェクト（アイコン）とその関係表現を基本とするものである。基本的な処理単位をアイコンとして用意しておき，画面上でアイコンとアイコンを結ぶことによってプログラムを作成していく。

以上のビジュアルプログラミング技術のうち，視覚的ユーザインタフェースと視覚的編集はプログラミング言語の概念から開放されていないので，エンドユーザコンピューティングという観点では，不十分であろう。視覚的言語に関しては，プログラミング言語の概念は不要であるが，プログラミングの概念は必要である。

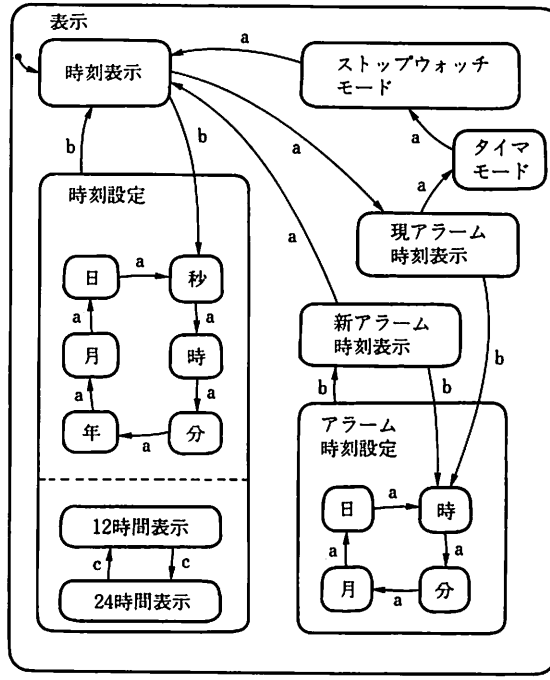


図 11.3 Statechart の例（ディジタル腕時計の仕様の一部）

11.4　第4世代言語

エンドユーザコンピューティングを指向した業務向け簡易言語として第4世代言語（4GL）がある。第4世代言語という名前は，第1世代言語（機械語），第2世代言語（アセンブラ），第3世代言語（コンパイラ言語）に続く言語という意味で付けられた。James Martinは，4GLを13ヶ条の特徴で定義したが，その主なものは，非職業的プログラマが使用可能，アプリケーションプログラムの記述ステップ数と作成工数がCOBOLより一桁少ない，非手続き的記述形式の採用，などである。

4GLの普及は米国が進んでいるが，現在，日本では数十種類の4GLが流通していると思われる。4GLの現状調査によると，4GLユーザの98%がエンドユーザだけでのソフトウェア開発可能性を否定している。その理由は，現在の4GLが，COBOLと比較した展開率の向上という「量的高級化」アプローチに基づいて設計されており，プログラミング技術を必要としているためと思われる。

今後，4GLのような業務向け簡易言語をエンドユーザコンピューティングのツールとしていくためには，業務の言葉で要求仕様を記述でき，業務知識のデータベースを用いてプログラムを自動生成できるようにする必要がある。