4D－5

オブジェクト指向プロトタイピングのための視覚的支援環境 ―関係定義とその支援ツール―

長崎武司*　阿部倫之*　中川秀敏**
金沢工業大学*　金沢工業高等専門学校**

１はじめに

オブジェクト指向を用いたプロトタイピングでは、オブジェクトの生成と消去、およびオブジェクト間の関係付けと、その解消を迅速に行えることが必要である。オブジェクト間の関係付けには、OMT の関連のように分析段階において宣言的に行われるが、実装時にはオブジェクト内に名前やポインタなどを埋め込むことによって代用されるのが一般的である。したがって、オブジェクトの連動を実現する場合、関係するオブジェクトへの名前やポインタをあらかじめ格納しておき、必要のつど検索をして対象のメソッドを呼び出す。すなわち、分析段階で宣言的に行われた関係記述が実装段階では手続き的に行われる。ギャップの解消や理解性を考慮すると実装段階でも宣言的に行うことが重要と考える。

本稿では、実装段階でオブジェクト間の関係付けを宣言的に行えるようにしたオブジェクト指向プロトタイピング支援ツールについて述べる。

２プロトタイピング方式

本プロトタイピング方式では、実世界に存在する実体(entity) を、何らかの知覚できる表現 (presentation) と能力 (ability) を持つものとして捉え、実体をこの２つのプリミティブで構成する。この２つのプリミティブをそれぞれ視覚オブジェクト (view object) 、機能オブジェクト (function object) と呼び、この２つをカプセル化（一体化）したものをフレームと呼ぶ。実装の際には、フレームを管理するためのオブジェクト（フレームオブジェクト）を定義して、視覚、機能、フレームオブジェクトの３つを貼り合わせることによってフレームを構成する。このフレームは実体に１対１に対応しており、最終的には、フレーム同士を結合することによって、実行プロトタイプを構成する。

３オブジェクトの関係定義

機能オブジェクト、視覚オブジェクト、そしてフレームオブジェクトを貼り合わせてフレームを構成するが、このとき互いの状態変化を相互に反映させるための関係定義を行う。この関係定義には、(1)オブジェクトの貼り合わせ、(2)操作の貼り合わせ、(3)属性の貼り合わせの３つの方法がある。

３．１オブジェクトの貼り合わせ

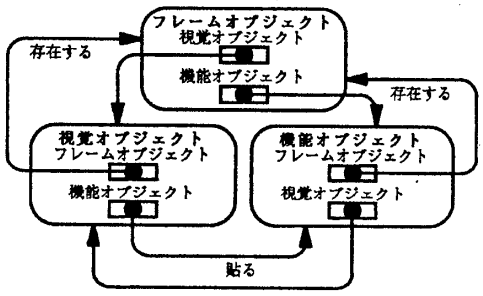

図１　ポインタによる関係定義例

フレームを構成する３つのオブジェクトを、ポインタを直接用いて関係定義をした場合、図１に示すようになる。

これを本支援ツールでは、図２に示すように、Prolog のホーン節を用いて宣言的に関係定義できるようにした。この結果、例えばフレームオブジェクトに関係している視覚オブジェクトを獲得する場合は、次のような目標を与えることで、

```
(?- (存在する (視覚 ?x FrameObject)))
```

関係オブジェクトを？x の値として獲得することができる。ここで？x などのように？で始まる（?-は除く）記号は変数を表している。

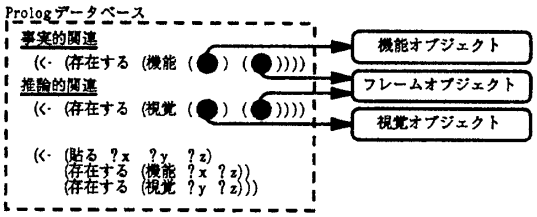

図２ 宣言的な関係定義例

また、図１の視覚オブジェクトと機能オブジェクト間の関連「貼る」に対するリンクを求める場合、図２に示すような推論的関連を定義すると、例えば次のような目標、

```
(?- (貼る FunctionObject ?x ?y))
```

を与えることによって、機能オブジェクトに関係するオブジェクトを？x、？y の値として導出することができる。すなわち、この推論的関連によってリンクの削減が可能となる。

３．２操作の貼り合わせ

操作（基本操作）の貼り合わせでは、貼り合わされた操作が連動するようなメカニズムを提供する。この貼り合わせには事前操作としての貼り合わせと、事後操作としての貼り合わせがある。この事前操作と事後操作は、基本操作に対するデーモンとして生成し、そのデーモン内に、“貼り合わせた基本操作”の呼び出しを記述する。

例として、図３で示すような電話のベルがなる場合について考える。まず、ベルが鳴るときは機能的な部分でベルのスイッチが ON になり、その後ベルが鳴ることから機能オブジェクトの「ベルのスイッチを ON にする」というメソッドの事後操作として、視覚オブジェクトの「ベルが鳴る」を指定する。この場合、「ベルのスイッチを ON にする」メソッドの事後操作デーモン（after デーモン）内で、

```
(?- (貼る FunctionTelephonObject ?x ?y))
```

とすると、先に述べたオブジェクトの貼り合わせにおいて関係を定義しているので、それをもとにして電話の視覚オブジェクトを導出し、機能オブジェクトのメソッドから電話の視覚オブジェクトの「ベルが鳴る」メソッドを呼び出すことが可能となる。

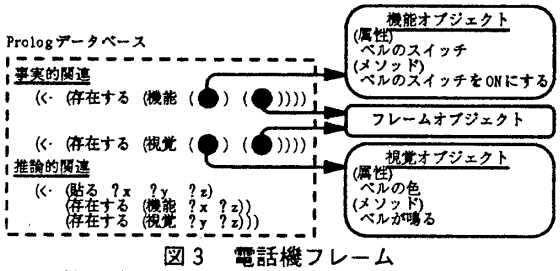

図３　電話機フレーム

３．３属性の貼り合わせ

属性の貼り合わせでは、属性のアクセス時に対応する属性値が変化するメカニズムを提供する。これは、属性のアクセス時に一方の属性値が書き換えられた時に、対応するもう一方の属性値が連動して変化するように writer デーモンを生成する。このデーモンは対応づけられた属性値にそれぞれ存在しており属性値の変化は双方向に行うことが

Visual Environment for Object-Oriented Prototyping.
Takeshi NAGASAKI,Noriyuki ABE
Kanazawa Institute of Technology

できる。

例として、節3.2と同様に電話機のベルが鳴る場合について考えると、機能オブジェクトの属性でベルの状態を表す「ベルのスイッチ」と視覚オブジェクトの属性でベルの状態を表す「ベルの色」が貼り合わされていたとする。ベルのスイッチがOFFからONになったときに連動してベルが鳴るためにベルの色を白から黒に書き換える。ここで、視覚オブジェクトの「ベルの色」と電話の絵が対応づけられていれば、図4に示すように、機能オブジェクトの「ベルのスイッチ」の変化に対応して電話の絵のベルの状態を変化させることが可能となる。属性の変換処理は、writerデーモン内に記述する。

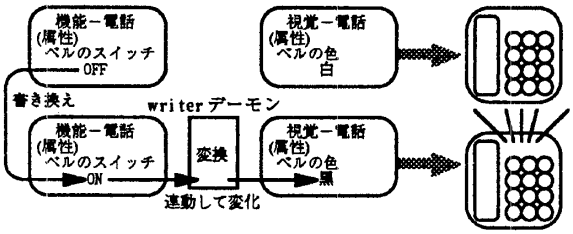

図4　属性の貼り合わせ

4 視覚的支援環境

4.1 システム構成

先に述べたプロトタイプ方式とオブジェクト間の関係定義方法を用いてプロトタイプの作成を支援するためのシステムを試作した（図5）。システムは5つのモジュールから構成されており、CLOS(Common Lisp Object System)を用いて、Macintosh上にインプリメントした。支援ツールは、オブジェクトをアプリケーションの視覚的なイメージを表現するものと、機能を表現するものに分離した設計環境を提供する。機能オブジェクトの設計は、クラス図エディタを用いてOMTのクラス図テンプレートに直接記述することで定義し、その修正やメソッドの定義については、テキストエディタで行う。視覚オブジェクトの設計は図形エディタにおいて視覚オブジェクトの目に見える部分である絵（実体図形）を作成し、この絵の情報をクラス図エディタに取り込み、OMTのクラス図テンプレートを用いてクラス化する。その修正やメソッドの定義については、機能オブジェクトと同様にテキストエディタで行う。次に、フレームエディタを用いて機能クラスとの貼り合わせを行いフレーム化する。このとき、クラス間の関係を定義し、また、操作の貼り合わせと属性の貼り合わせを行うと、デーモンのテンプレートが自動生成される。定義されたフレームを部品化し、プロトタイピングシェルにおいてフレームの表示（アイコン化）、フレームオブジェクトの生成、編集などを行い、実行可能なプロトタイプの作成、実行時におけるフレーム属性の参照などを実現する。

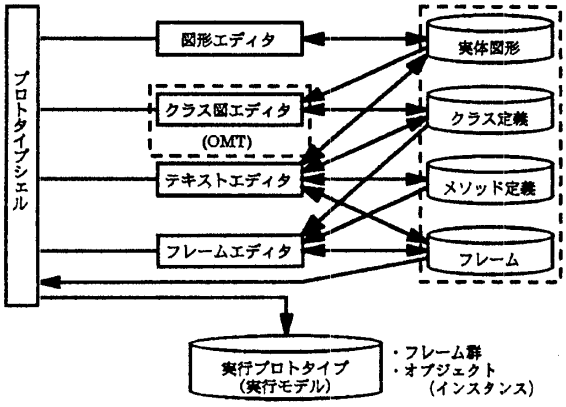

図5　システム構成

4.2 プロトタイピングシェル

プロトタイピングシェルは、各エディタおよび各プロダクトを取りまとめている。例として、図6のような簡単な電話交換システムのサービスプロトタイプの作成を用いて、プロトタイピングシェルの主な機能を以下に示す。

(1)フレームの編集

図形、クラス図、テキスト、フレームエディタを使用してフレームの編集を行う。フレームエディタでは、各クラス間、およびフレーム間の関係定義を行う。

(2)フレームの表示

フレームエディタにおいて、作成されたフレームをファイルから読み込み、部品ウィンドウ（FRAME-POOL）に登録することにより実体図形が表示される（フレームのアイコン化）。図6は、部品ウィンドウ（FRAME-POOL）に回線、交換機、電話機を登録し、3者間通話のプロトタイプを作成した例である。

(3)モデルの編集

部品ウィンドウ（FRAME-POOL）のアイコン（フレーム）をプロトタイプウィンドウ（PROTOTYPE-WINDOW）へドラッグ&ドロップすることにより、フレームクラスのインスタンスとフレームを構成するクラスのインスタンスの生成が行われ、実行可能なオブジェクトによるプロトタイプが作成される（図6）。

(4)シェル操作との連動

モデル生成などのシェル操作に対してユーザ操作を貼り付けることで、シェルのカスタマイズが可能である。図6の例では、回線フレームをプロトタイプ作成に使用しており、この時、回線はモデル生成操作に連動して実体図形を変形させている。また、この時フレーム間の接続も行っている。このような処理の連動は、CLOSのメソッド結合機能よって実現している[3]。

(5)プロトタイプモデルの実行

作成したプロトタイプモデル上で、実行メソッド名を指定することでオブジェクトを起動する。また、その時のモデルの属性を参照し、状態の変化を見ることができる。

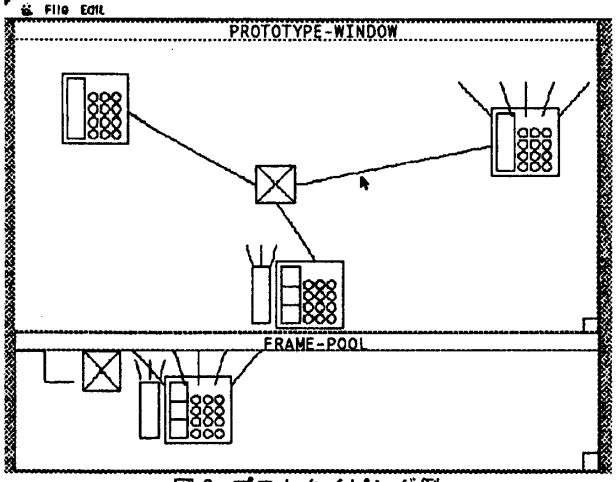

図6 プロトタイピング例

5 おわりに

本稿では、オブジェクト間の関係定義を宣言的に行う方法について述べ、また、それを用いたオブジェクト指向プロトタイピング支援ツールについて述べた。今後、分散オブジェクト指向環境への適用を検討していく予定である。

参考文献

[1]阿部倫之、他：オブジェクト指向プロトタイピングのための視覚的支援環境，情報処理学会ソフトウェア工学研究会，SE99-18，1994

[2]阿部倫之、他：交換サービスモデリングシステムの一検討，電子情報通信処理学会交換システム研究会，SSE90-114，1991

[3]湯浦克彦、高橋久：ODETTE：オブジェクト指向CLOSをベースとした設計支援構築環境，「オブジェクト指向ソフトウェア技術」シンポジウム，1991