

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

阿部倫之* 吉田正弘* 中川秀敏**

*金沢工業大学

**金沢工業高等専門学校

〒921 石川県金沢南郵便局区内野々市町扇が丘7-1

E-mail: abe@infor.kanazawa-it.ac.jp

本稿では、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について述べる。支援環境では、視覚オブジェクトと機能オブジェクトを貼り合わせたフレームを作成し、このフレームを結合することでプロトタイプシステムを構成する。さらにフレームをアイコンとして表示し、オブジェクトの参照や実行をアイコン操作によって実現するビジュアルな設計環境の実装法を示す。また、オブジェクトのクラス構成に制約と関連を加えることで、オブジェクト間の関係やオブジェクト自身の制約条件の宣言的な記述が可能となった。この基本概念と実装法についても述べる。

A Visual Environment for Object-Oriented Prototyping

Noriyuki ABE, Masahiro YOSHIDA and Hidetoshi NAKAGAWA

Kanazawa Institute of Technology

Kanazawa Institute of College

7-1, Ohgigaoka, Nonoichi-machi, Kanazawa-south area, Ishikawa, 921, Japan

E-mail: abe@infor.kanazawa-it.ac.jp

In this paper, we describe a visual environment for object-oriented prototyping. A useful application of this tool is in defining frames that are constructed with visual objects and functional objects. A prototype system is constructed with frames that are represented by icons. We show the implementation method of visual environment that the functional object can be driven by manipulating icons. In addition, we show the method for defining the relationship of objects by applying constraint and association to the prototyping method.

1 まえがき

ソフトウェア開発の初期段階では、対象としている製品の概念的構造とその概念的動作を決定し、制約等も含めて要求仕様としてまとめていく。この段階において、実行可能な仕様を試行プログラミングによって作成し、概念構造と動作の確認や検証を行い、修正をリアルタイムに実施していくプロトタイピングが注目されている。プロトタイピングには、拡張や修正が頻繁に行われることから、概念をオブジェクトで表現し、この作業をビジュアルに支援する環境が検討されている。ビジュアルプロトタイピングとしては、アイコン化された視覚部品の組み合わせを主操作とする支援環境が提案されている(1)(2)。これらアイコン操作を主体にした方法は、エンドユーザを含めて容易に操作、カスタマイズできるが、新機能を持つアイコンの設計など、組み合わせ操作の範囲内で実現できない場合には、何らかの(視覚)言語を用いたプログラミングを必要としている。

ここで、プログラミングの熟練者の立場からビジュアルプロトタイピングを考えた場合、プログラミング(テキストまたは図式記法による)を主にして、GUIなどの直感的に作業したい部分や動作検証をアニメ表示などを用いてビジュアルに支援するのみで十分な場合も多い(LISPのような記号処理向き言語を使用している場合など)。この考え方による支援環境として、湯浦氏らの開発した、CLOS(Common Lisp Object System)ベースの視覚プログラミング環境ODETTE⁽³⁾がある。ODETTEでは、まず、テキストベースでオブジェクト指向プログラミングを行い機能オブジェクトを作成する。次に、ビジュアル表示を必要としている機能オブジェクトに対しては、オブジェクト指向図形エディタによって視覚オブジェクト(ODETTEでは表示オブジェクト)を作成し、結合して連動させるメカニズムを提供している。このODETTEを用いて、構内交換機のサービス仕様の一部をプロトタイピングし、仕様の検証やプレゼンテーション等に適用してその効果を確認している⁽⁴⁾⁽⁵⁾。

本稿で述べるビジュアルプロトタイピング環境は、ODETTE流の考え方を基本にしており、まずオブジェクト指向プロトタイピング方式の基本構想について述べる。次に、制約とOMT流の図式記法(特に関連)を直接使用すること、および機能オブジェクトと視覚オブジェクトの合成について、具体的な支援ツールの構成を示しながら述べる。

2 プロトタイピング方式

2.1 機能オブジェクトと視覚オブジェクト

実世界に存在する実体(entity)を、何らかの知覚できる表現(identity)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および機能オブジェクトで実現し、実体に1対1に対応したフレームとしてカプセル化する(図1)。このフレームを結合することによって機能合成を行い、最終的な実行ビジュアルプロトタイプを構成する(図2)。

プロトタイピングの基本手順を以下に示す。

- (1) 機能オブジェクトのみによる実行可能プロトタイプを作成する。
- (2) 機能オブジェクトに対応した視覚オブジェクトを実体図形を用いて直接作成する。
- (3) 機能オブジェクトと視覚オブジェクトを貼り合わせてフレームを作成する。このとき操作の連動と属性の同化を行い機能合成する。
- (4) フレームを結合して実行可能ビジュアルプロトタイプを作成する。

ここで、フレーム間の機能合成については、機能オブジェクトのみを用いたプロトタイプ作成によってなされている(熟練プログラマの能力に依存している)ため、支援ツールとしては、オブジェクト間のリンクを生成するのみでよい。従って、重要となるのは、視覚オブジェクトと機能オブジェクトを連動させるための機能合成であり、主要な支援の対象としている。

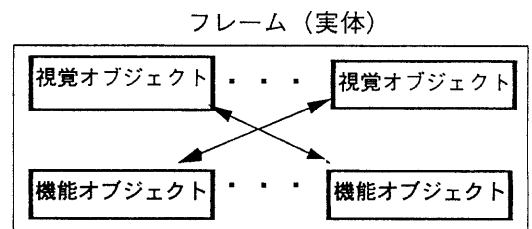


図1 フレームの構成

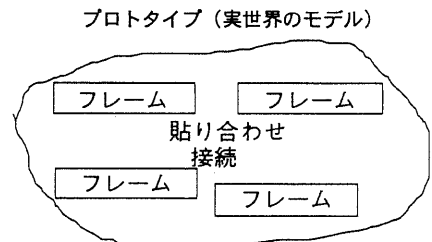


図2 プロトタイプの構成

2.2 オブジェクトの構造

オブジェクトの構造を定義するクラスは、一般に、クラス名、属性、操作の3つの要素で構成される。本プロトタイピングでは、拡張性、再利用性、検証性を考慮して、制約とオブジェクト間の関連をクラス内に直接記述する(図3)。制約には、クラスの制約と操作の制約があり、操作の制約としては、事前条件と事後条件が記述できる。関連は、クラス間の関係を図式記法で明示するためのものであり(6)(8)、通常、実装時にはポインタとして手続的に扱われるが、これを宣言的に扱えるように検討している。具体的には、関連および制約を述語論理の節形式を用いてクラス内に記述し、オブジェクト間の関係の宣言や、関係の導出が実装時にできるよう工夫している。

名前
属性
操作
制約 クラスの制約 操作の制約 (事前条件、事後条件)
関連 事実的関連 推論的関連

図3 拡張クラスの構成

2.3 機能合成

機能オブジェクトと視覚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機能合成を行う。合成には、以下の3つの方法を用いる(図4)。

(1) オブジェクトの貼り合わせ

機能オブジェクトと視覚オブジェクトを結合するためのリンクを生成(事実的関連のインスタンス)。

(2) 操作の貼り合わせ

機能オブジェクトと視覚オブジェクトの操作に対して、事前操作および事後操作をデモンとし生成し、その処理内に連動操作呼出しを記述する。

(3) 属性の貼り合わせ

機能オブジェクトの属性と視覚オブジェクトの属性において、属性のアクセス時に、一方の属性が書き換えられた場合に、対応するもう一方の属性の値がそれに連動して変化するように、ライトデモンを生成する。このデモンは、対応付けされた属性に直接アク

セスすることが可能であり、それぞれ独立して対になって存在している。

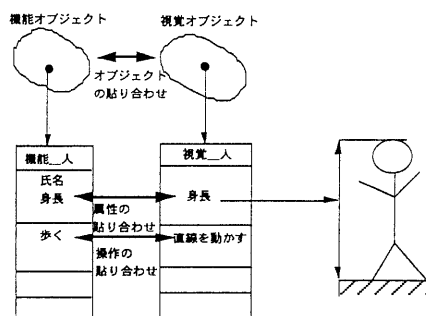


図4 機能合成

3 制約と関連

3.1 制約

オブジェクトの制約条件を、宣言的に記述することにより、次の効果を期待している。

- (1) 手続的に記述されていたオブジェクトの性質を、論理式を用いて直感的に理解できる形式で明確に定義できる。
 - (2) 操作中からオブジェクトの存在条件に関する処理を分離でき、修正や拡張の影響を減少できる。
- 制約をクラスの制約と操作の制約(事前条件、事後条件)に大別する。

クラスの制約

クラスの制約は、オブジェクトが常に満たすべき条件であり、この条件によりオブジェクトの動作が保証される。属性間の関係の記述が主であり、図5のスタックの場合、スタックポインタの性質をクラスの制約として記述している。制約条件が満たされない時は、該当オブジェクトと不成立制約が付加されたメッセージを発行する。

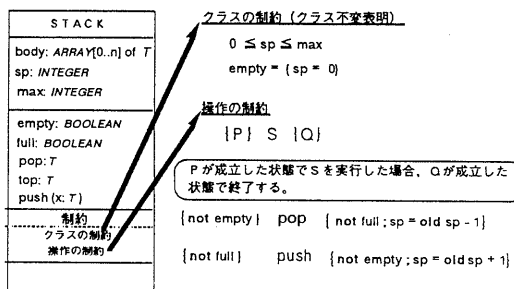


図5 制約の例

操作の制約

操作の制約は、オブジェクトに所属する操作に付加する動作条件である。事前条件は、操作を実行する前に成立していなければならない条件、事後条件は、操作の実行後に成立していなければならない条件である。図5の例では、スタックのpush, pop操作に対して操作の制約を付加している。制約内容は、属性間の関係記述が主体となる。また、事後条件の例の場合、制約を手続き的に解釈すると、スタックポインタの変更処理をすることになり、操作の一部を制約として宣言的に記述できる。

制約の評価

制約の評価は、操作が呼び出されることで起動され、Prologのゴール節と同様の形式で評価される。評価の順序は、操作の実行前に、

(1) クラスの制約

(2) 操作の事前条件

の評価を行い、操作の実行後に、

(3) 操作の事後条件

(4) クラスの制約

の評価を行う。

3.2 関連

OMTにおける関連は、オブジェクト間の関係を表したものであり、図6のようなクラス図中において、クラス間を結ぶアークで表現される。この例は、大学に学生として存在している人と教官として存在している人がおり、教官は学生を教授する関係にあることを表現している。関連は、クラスのインスタンスであるオブジェクト間ではリンクと呼ばれる。一般にこのリンクは、ポインタとして実装され、オブジェクト内の属性化された関連名に格納される。従って、操作の過程において、関連に關係する処理にはポインタの検索や手繰りといった手続き的な記述がなされる。ここで、試行プログラミング段階におけるプロトタイピングでは、分析工程とのギャップをできるだけ少なくし、変換プロセスを減少させることが望ましい。そこで、この関連を述語論理の事実節として宣言的に実装し、リンクを事実節のインスタンスとして宣言的に表現する方法を検討している。

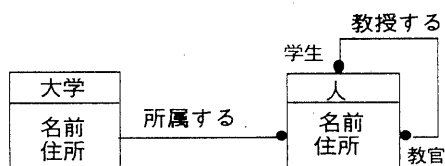
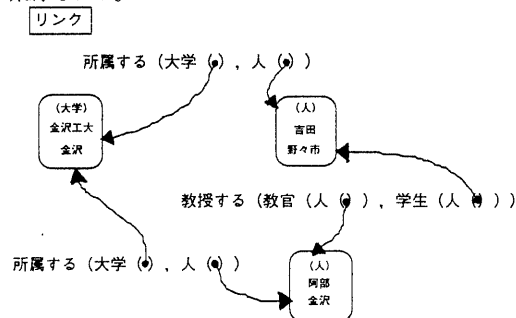


図6 関連の例

図7は、図6のインスタンスの例であり、リンクを宣言的に表現している。このような関連を事実的関連と呼ぶ。事実的関連は述語論理形式で記述しており、?で始まる記号は変項を表している。

述語名には関連名、関数名にはクラス名またはロール名が対応しており、変項にはオブジェクトへのポインタが束縛される。

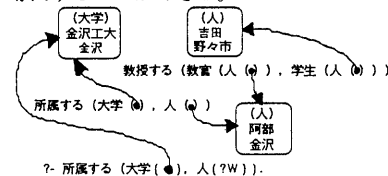


事実的関連

所属する (大学 (?X), 人 (?Y))
教授する (教官 (人 (?X)), 学生 (人 (?Y)))

図7 事実的関連とリンクの表現

図8は、関連を満足するようなオブジェクトを獲得する例である。?で始まるPrologのゴール節形式が与えられると処理が始まり、リンクとのユニフィケーションによって所定のオブジェクトが獲得される。この例では、金沢工大に所属する人の獲得を試みており、オブジェクト阿部のみが得られる。ここで、図8に示すような推論的関連を与えると、オブジェクト吉田も獲得できる。この推論的関連は、'ある人が大学に所属していて学生を教授しているならばその学生も大学に所属している'という規則を表しており、これによって新しいリンクを導出することができる。



推論的関連

所属する (大学 (?X), 人 (?Z)) ← 所属する (大学 (?X), 人 (?Y)) ∧
教授する (教官 (人 (?Y)), 学生 (人 (?Z)))

図8 リンクの導出

この方法は、関連を宣言的に解釈して扱うことができ、リンクをポインタで表現した場合に比べて関係の追加や変更が自然な形で実現できる。

4 視覚的支援環境

4.1 システム構成

支援環境のシステム構成を図9に示す。システムは、大きく5つの支援モジュールにより構成されており、CLOSを用いてMacintosh上にインプリメント中である。機能クラスの設計は、基本的にテキストベースで行うが、クラスのテンプレートレベルまでは、クラス図エディタが使用できる。従って、支援の中心は、視覚クラスの設計であり、事前に設計した機能クラスとの貼り合わせ（フレーム化）、フレーム表示、フレームのオブジェクト生成、編集などがある。各支援モジュールの詳細を次節以降で述べる。

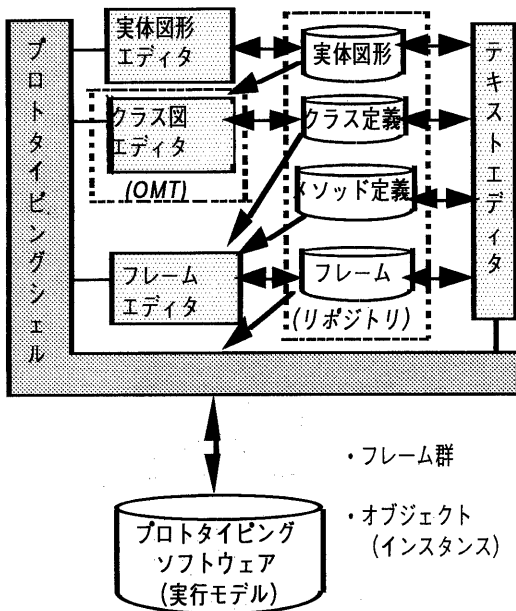


図9 システム構成

4.2 プロトタイプリングシェル

プロトタイプリングシェルは、実体図形エディタ等の支援モジュール群を取りまとめている。主な機能の概要を次に示す。

(1) フレームの編集

実体図形、クラス図、テキストエディタを起動し、

フレームの編集を支援する。各エディタからのプロダクトはエディタ自身が管理する。

(2) フレームのビジュアル表示

作成したフレームの実体図形を表示する。すなわち、この段階でフレームがアイコン化される。このアイコンを用いて、定義されている操作（メソッド）の参照や属性（スロット）の参照などが行える。

(3) モデルの編集

フレームのアイコンをプロトタイプ作成ウィンドウ（モデルウィンドウ）上に複写することで、フレーム内にあるクラスのインスタンス生成が行われ、実行可能なオブジェクトによるプロトタイプが作成できる。

(4) シェル操作との連動

複写（インスタンス生成）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10の例では、ワイヤフレームをプロトタイプモデルに使用しており、このとき、ワイヤは複写操作に連動して変形している。このような処理の連動は、CLOSのメソッド結合機能によって実現する。

(5) 制約と関連の評価

制約と関連はS式で宣言し、CLOSで記述されたPrologインタプリタを用いて評価する。制約はオブジェクト内に閉じて評価し、関連（リンク）は、実行プロトタイプ内に閉じて評価する。具体例については節4.3以降で示す。

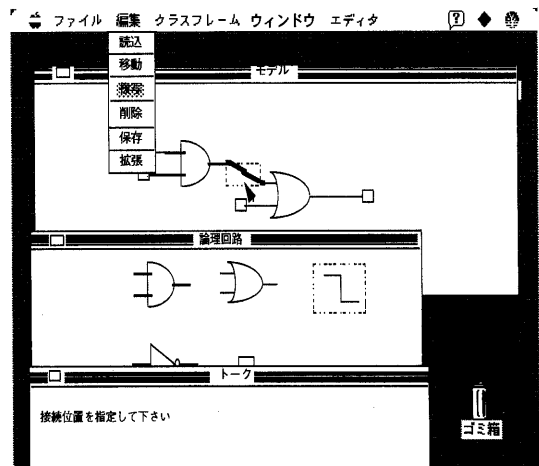


図10 表示画面例

4.3 図形エディタ

視覚オブジェクトを表現する実体図形を描画するためのエディタであり、実体図形は基本図形（円、楕円、円弧、長方形、多角形、直線）の組み合わせで構成される。基本図形と描画された複合図形は、以下のような描画スクリプトとして保存される。

基本図形（直線）例

(直線 (座標 1 (10 20)) (座標 2 (10 30))
(太さ 1) (色 赤) (線種 実線))

複合図形（AND）例

(AND (直線1) (直線2)
(直線3) (直線4)
(円弧1))

図11は、論理回路設計システムのプロトタイプینگに使用するAND素子フレームの実体図形（基本図形である直線4本と円弧1本から成る）を描画し、複合図形名ANDを付加した例である。

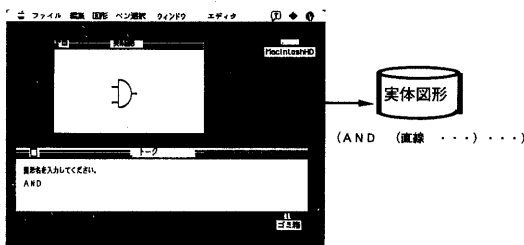


図11 図形エディタ画面例

4.4 クラス図エディタ

クラス図エディタは、オブジェクトのクラス設計を、OMTの図式記法で行うための支援ツールである。このツールを用いて機能クラスと視覚クラス的设计を行う。

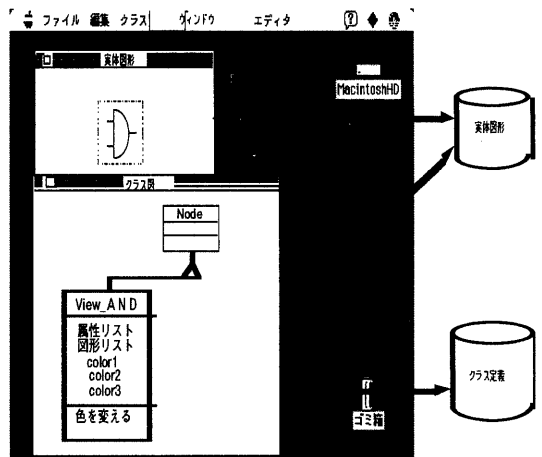
機能クラス的设计は、クラス名、属性、操作、制約、関連のフィールドをもつ機能クラステンプレート内に記入することでを行い、最終的にCLOSのクラス定義を生成する。この定義は、テキストエディタを用いて修正可能である。また、テキストエディタで定義の変更を行った場合は、クラス図に反映される。機能クラスANDの一部を以下に示す。

```
(defclass AND (素子)
  ((制約 : initform '(((電圧をかける ((?- (属性 状態 発火)))
                                         ((?- (属性 状態 発火))))
            (動作する ((?- (属性 状態 活性化))
                           ((?- (属性 状態 発火)))))))
```

```
(クラス ((?- (属性 出力 ?X)
             (属性 入力 1 ?Y)
             (属性 入力 2 ?Z)
             (is ?x (and ?Y ?Z))))))
(関連 : initform '(((接続する (入力 (素子 ?X))
                                   (出力 (ワイヤ ?Y))))
      ((?- (接続する (出力 (素子 ?X))
                      (入力 (ワイヤ ?Y))))))
```

制約はPrologのゴール節と規則節、関連は、事実節と規則節の形式が中心となる。このうち、ゴール節と事実節は直接評価の対象となり、規則節は評価の過程で用いられる。ゴールの評価結果は、真または偽であり、偽の場合、該当オブジェクト名と制約がメッセージとして表示される。

視覚クラスは、実体図形を参照しながら設計可能であり、クラス属性を用いて図形内の属性を変更するための簡単な手続きを提供している。図12に、ANDの視覚クラスView_ANDを作成する例を示す。図形エディタで描画したAND図形を実体図形ファイルから読み込み、View_ANDクラスの属性である図形リストの初期値として保持する。ここで、基本図形に識別子を付加し、基本図形の属性を視覚クラスの属性から変更できるように対応関係を付ける。この対応付けを属性リストとして視覚クラス内に保持し、視覚オブジェクトを表示 (redraw) する際に、クラスの属性値が図形の属性値として反映される。例えば、図12における、視覚クラスの属性color1、color2、color3は、それぞれ実体図形の入力線と出力線の色属性に対応付けされている。設計者が、色を変更する操作'色を変える'を実装する場合、自分で設定したcolor1からcolor3を書き換えればよく、図形の詳細なフォーマットを意識する必要がなくなる。



```
(defclass View_AND (node)
  ((属性リスト : initform '(((color1 (入力 1 色))
                              (color2 (入力 2 色))
                              (color3 (出力 色))))
   (図形リスト : initform ' (AND (入力1 (直線... (色 黒)...))
                                (入力2 (直線... (色 黒)...))
```

```

(出力 (直線・・・(色 黒)・・・))
((直線・・・(色 黒)・・・))
((円弧・・・(色 黒)・・・))

```

図12 クラス図エディタの画面例

4.5 フレームエディタ

フレームエディタは、機能クラスと視覚クラスを貼り合わせ、機能合成を支援する。その際に、フレームを管理するためのクラス（フレームクラス）を1つ定義する。機能合成の方法を次に示す。

(1) 機能クラスと視覚クラスの貼り合わせ

機能クラスと視覚クラスの貼り合わせとは、既定義の視覚クラスと機能クラスをまとめて1つのフレームとして定義することである。図13では、機能クラスANDと視覚クラスView_ANDを貼り合わせている。手順は、
 (i) フレームメニューのフレーム定義を選択
 (ii) 視覚、機能クラスの指定
 であり、AND_Frameがシステムにより自動生成される。さらにAND_Frameの関連フィールドには‘存在する’という関連が登録され、視覚、機能クラスには‘貼る’という関連がそれぞれの関連フィールドに登録される。これらの関連により、フレームは自分自身が管理すべきクラスが存在を知り、各クラスは自分自身が対応しているクラスを意識することができる。

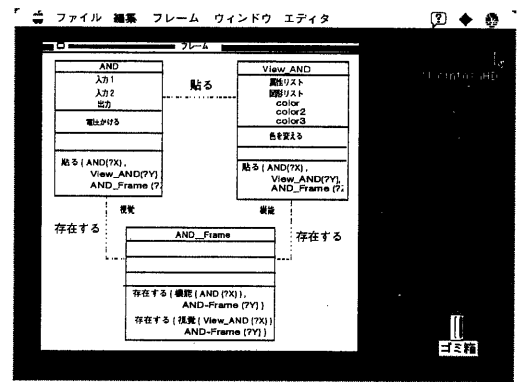


図13 フレーム定義

(2) 操作の貼り合わせ

視覚クラスと機能クラスの間接的な操作を連動させる場合に操作の貼り合わせを行う。図14では、機能クラスの‘電圧をかける’という操作に連動して視覚クラスの‘色を変える’という操作を呼び出すために、貼り合わせ処理を行っている。操作手順は、

- (i) フレームメニューの操作の貼り合わせを選択
 - (ii) 貼り合わせたい操作をそれぞれ指定
- であり、貼り合わせデモン（操作の連動処理を記述）としてシステムが自動生成する。また、貼り合わせた操作やデモンを、テキストエディタを用いて修正することも可能である。

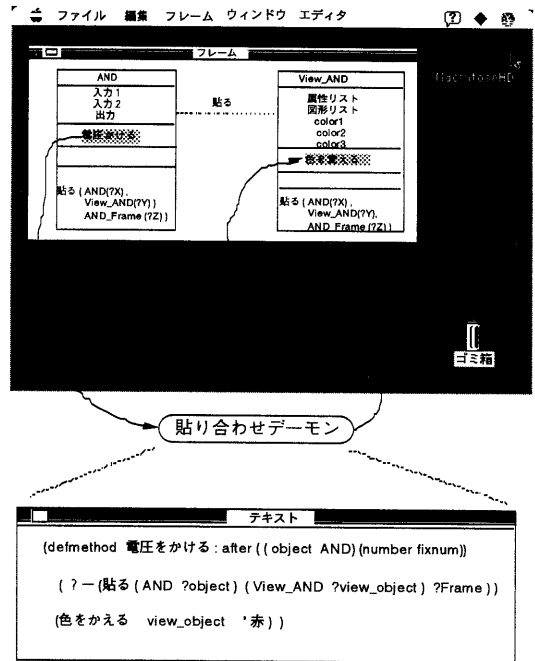


図14 操作の貼り合わせ

(3) 属性の貼り合わせ

属性の貼り合わせは、視覚クラスの属性と機能クラスの属性において、属性のアクセス時に一方の属性値が更新された場合に、対応するもう一方の属性値もそれに連動して変化するように設定することである。図15ではANDクラスの入力1とView_ANDクラスのcolor1の貼り合わせをクラス図上でやっている。操作手順は、
 (i) フレームメニューの属性の貼り合わせを選択
 (ii) 貼り合わせたい属性をそれぞれ指定
 であり、貼り合わせデモンはシステムが自動生成する。また、自動生成されたデモンは、テキストエディタを用いて修正できる。

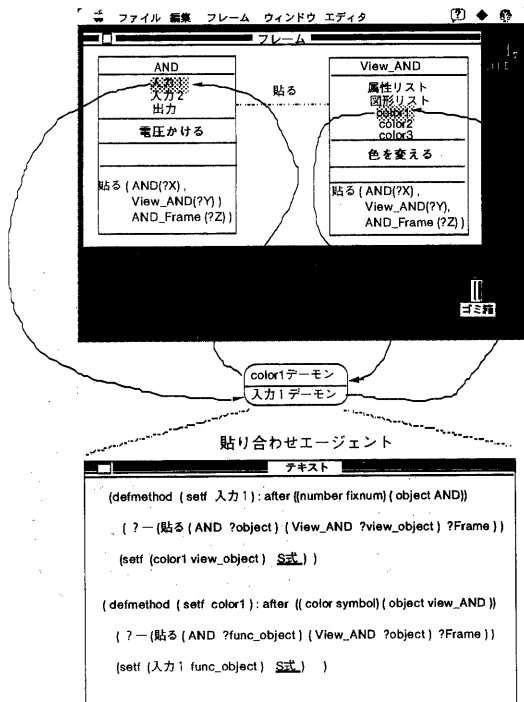


図15 属性の貼り合わせ

5 おわりに

本稿では、オブジェクト指向プロトタイピングのための視覚支援環境について、基本構想と支援ツールの構成について述べた。テキスト（図式記法）ベースのプログラミングから出発して、必要に応じてオブジェクト単位のビジュアル化を図る専門家向きではあるが、アイコン化されたフレームの組み合わせについては、非専門家でも容易に操作できる。同様の考え方をとるODETTE上で交換機のサービス仕様のプロトタイピングを行い、操作の容易さについては確認している⁽⁴⁾⁽⁵⁾。今回の支援ツールでは、さらに、機能オブジェクトのプログラミングに図式記法を導入しているのと、関連と制約の宣言的記述、およびオブジェクトの貼り合わせを支援しており、プロトタイピング効率の向上が期待できる。

現在、支援ツールは、Macintosh上にCLOSを用いて作成中であり、今後具体的な応用事例に適用して評価を進める予定である。

謝辞

本研究の一部は、株式会社 日立製作所情報通信事業部ソフト技術センタ殿の委託研究の一環として行われたものである。

参考文献

- (1) 牧村信之，坂本浩一：情報の高度利用を目標にソフトの部品化・再利用を実現する「IntelligentPad」，『オブジェクト指向技術総覧'93-'94年版P258-275，日経BP社，1993
- (2) 鳥居明彦，加登基二，平川正人，市川忠男：オフィス向けH I-V I S U A L システム，情報処理学会ソフトウェア工学研究会，SE97-9，1994
- (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
- (4) 阿部倫之，前島幸仁，湯浦克彦，丹野千秋：交換サービスモデリングシステムの一検討，電子情報通信学会交換システム研究会，SSE90-114，1991
- (5) 前島幸仁，阿部倫之，湯浦克彦：オブジェクト指向による交換サービス仕様記述について，第2回N Aワークショップ論文集，C-2，1992
- (6) B.Meyer：OBJECT-ORIENTED SOFTWARE CONSTRUCTION，二木厚吉 訳：オブジェクト指向入門，アスキー出版局，1990
- (7) Nan C.Shu：VISUAL PROGRAMING，西川 訳：ビジュアルプログラミング，日経BP社
- (8) J.Rumbaugh：OBJECT-ORIENTED MODELING AND DESIGN，羽生田栄一 訳：オブジェクト指向方法論，トッパン，1992
- (9) S.Shlaer，S.J.Mellor：Object-oriented System Analysis：Modeling the World in Data，Prentice-Hall，1988
- (10) P.Coad，E.Yourdon：Object-Oriented Analysis，Prentice-Hall，1990