4M-6

オブジェクト指向 UIMS CONCORD (1)
—共有 ADT メカニズム—

間宮 悟, 内藤広志, 松山洋一, 浅野俊昭
キヤノン(株) 情報システム研究所

1 はじめに

近年ダイレクト・マニピュレーションがユーザインタフェースとして標準となりつつある. このダイレクト・マニピュレーションを UIMS (User Interface Management System) で実現しようとする時, ユーザインタフェースをアプリケーションから完全に分離させるという, UIMS の基本的な考え方の見直しが求められている[1][2][3].

ダイレクト・マニピュレーションにおいては, ユーザ操作に伴うアプリケーションの状態変化の表示や, 操作の有効性のユーザへの通知といったようなsemantic feedback がたびたび必要となる[4]. ところが, この様な機能を実現するには, データの表示というユーザインタフェースの仕事にアプリケーションに固有な情報が頻繁に必要となる. ユーザインタフェースとアプリケーションを分離してこのような feedback を実現するには, アプリケーションの持つデータ構造をユーザインタフェースでも二重に持たなければならなくなる. すると両者の整合性の維持等が問題となってくる.

そこで我々はダイレクト・マニピュレーションをサポートする際に, 従来の UIMS の持つ利点を生かしつつ上記の問題を解決するための, ユーザインタフェース/アプリケーション・メカニズム—**共有 ADT メカニズム**— を採用し, このメカニズムに基づいて現在オブジェクト指向 UIMS **CONCORD** を 開発している.

2 共有 ADT メカニズム

ユーザインタフェースはアプリケーションが操作するデータ(アプリケーション・オブジェクト)を何らかの形態で表示するが, その表示形態を実現するのに必要な情報には二種類ある. 一つは個々のアプリケーション・オブジェクトを表示するためのデータであり, もうひとつはアプリケーション・オブジェクト間の関係を示すデータ構造である. 例えば, オブジェクト・クラスの階層を表示するようなユーザインタフェースにおいて, 前者は各クラスの名前を示す文字列であり, 後者はクラスの階層関係である.

両者のうち, データ構造はアプリケーション側にも存在するものであるから, 各々で二重に持たずに共有することができる. そこで **CONCORD** では, ユーザインタフェースとアプリケーションでデータ構造を表現する **ADT** (Abstract Data Type) を共有するメカニズム—**共有 ADT メカニズム**— を採用した.

Object-Oriented UIMS CONCORD (1) —Shared ADT mechanism—
Satoru MAMIYA, Hiroshi NAITO, Youichi MATSUYAMA, Toshiaki ASANO
Information Systems Research Center, Canon Inc.

2.1 構成オブジェクト

共有 ADT メカニズムは, 以下の 3 種類のオブジェクトにより構成される.

ADT ユーザインタフェースとアプリケーションで共有される抽象化されたデータ構造を持つオブジェクトで, データ構造を管理する責任を負う. データ構造を構成する各要素がアプリケーション・オブジェクトとなる.

ビュー ユーザに対する入出力を管理するオブジェクトで, 特定のアプリケーションに依存せず汎用性を持っている. また, 視点(point of view)という属性を持つ.

モデル ビューに対する ADT の受渡し, 表示の通知, 複数のビュー間の調整といったようなビューの管理をするオブジェクト. ビューを組み合わせて管理をする事によって, アプリケーション固有のユーザインタフェースを実現する.

3 つのオブジェクトは 図 1 の様に相互に結合されている. ビューはモデルと ADT をインスタンス変数に保持する. 一方, モデルは管理するすべてのビューを依存リストに保持する. この結合関係により, いわゆるプラガブルビュー[5] が実現される.

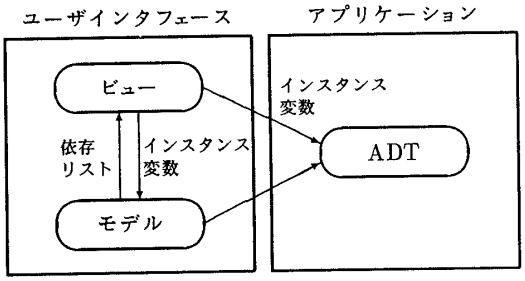

図 1: オブジェクトの結合関係

2.2 オブジェクト及びアプリケーション間のプロトコル

各オブジェクトとアプリケーション間のプロトコルは次のとおりである.

アプリケーションから ADT アプリケーションはデータ構造に関する操作を行なう際に ADT の手続きを呼ぶ. ADT は構造に関する処理を行なう. 構造操作に伴うアプリケーション固有の処理(各要素の属性の変更, エラーチェック等)は, アプリケーションが登録した hook により行なわれる.

ビューからモデル モデルは ADT を得るための手続き(あるいはメソッド)を用意し, ビューはインスタンス変数にその名前を持ちそれを呼ぶ.

モデルからビュー ビューは表示をするためのメソッドを用意する．モデルは表示の必要な視点を持つビューに対して，依存リストを通じてそのメッセージを送る．

ビューから ADT ビューは，表示データやデータ構造を得たり，データ構造を操作したりするために ADT の手続き（あるいはメソッド）を呼ぶ．表示データの取得等の，アプリケーションに固有な処理は ADT に登録されたhookにより実行される．

ユーザインタフェースであるモデルとビューは，アプリケーションが用意している ADT にプロトコルのみを介して結合されるので，インタフェースが変更されてもアプリケーションの変更は必要ない．

3 共有 ADT メカニズムの動作例

3.1 ユーザからの指示による動作

共有 ADT メカニズムの動作例として，ユーザがビューによってあるアプリケーション・オブジェクトの削除を指示した場合を，図 2 と以下に示す．なお，図中の番号は説明の番号に一致している．

1. ユーザがビューに表示されているアプリケーション・オブジェクトの削除を指示．
2. ビューがそのオブジェクトを引数とした ADT の要素削除手続きを呼ぶ．
3. ADT がオブジェクトを削除する．手続きの hook に登録されているアプリケーション固有の処理もこのとき行なう．アプリケーションの事情によりオブジェクトが削除できない場合，ADT の削除手続きはビューにエラー値を返す．
4. ADT から返ってきた値が
   (a) 正常値ならば，ビューはモデルにデータが変更されたことを伝える．
   (b) エラー値ならば，エラーをユーザに表示する．
5. モデルが依存リストを通じて，関連する視点を持つビューに表示を通知する．

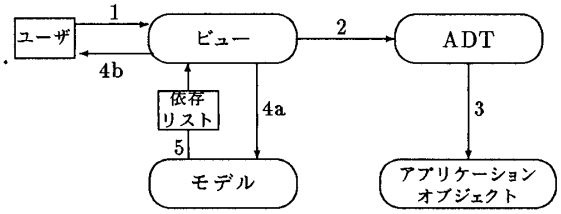

図 2: ユーザの指示による動作

共有 ADT メカニズムでは上記のように，ビューが直接 ADT の手続きを呼ぶので，ユーザ入力による操作がそのままアプリケーション・オブジェクトの操作になる．よって，データ構造を重複して持つことからくる整合性の維持の問題が解決され，すばやい semantic feedback の実現が期待される．

3.2 ビューの表示動作

表示を通知されたビューは以下のように表示を行なう．

1. モデルから ADT を得る．手続き（メッセージ）名はビューのインスタンス変数に登録されている．
2. ADT から各々の要素の表示データを得る．
3. ADT からデータ構造を得る．
4. 表示を行なう．

4 CONCORD の構成

我々は現在 **ADT メカニズム**に基づいて Common Lisp と Flavors 上でX-Window を用いたユーザインタフェースを作成するためのオブジェクト指向UIMS **CONCORD** を開発している．**CONCORD** は，**note**（flavor のインスタンス）という部品を基本要素とし，図 3 の様に，**note** の基本機能を実現する**Intrinsics** と，ライブラリである **Note-lib** から構成されている．ビューはこの **note** を組み合わせることにより実現される．

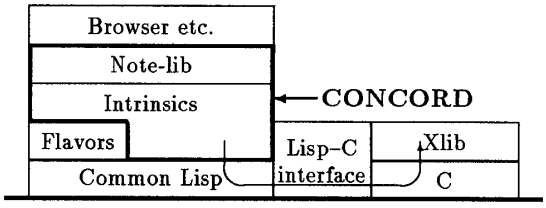

図 3: **CONCORD** の構成図

5 おわりに

これまでに述べてきたように **共有 ADT メカニズム**には以下の特長がある．

- ユーザインタフェースとアプリケーションが ADT のプロトコルを介してのみ結合されているので，互いのモジュラリティーが保証される．
- ユーザインタフェースとアプリケーションでデータ構造を共有するため，効果的な semantic feedback の実現が期待される．

参考文献

[1] Dance J. et al The Run-time Structure of UIMS-Supported Applications. *Computer Graphics 21*, 2 (April 1987), 97-105.

[2] Löwgren, J. History, State and Future of User Interface Management Systems. *SIGCHI Bulletin 20*, 1 (July 1988), 32-44.

[3] Rosenberg, J. et al UIMSs: Threat or Menace? *Proceedings of the CHI'88 Conference*, 197-200.

[4] Hudson, S. UIMS Support for Direct Manipulation Interfaces. *Computer Graphics 21*, 2 (April 1987), 120-124.

[5] 上谷晃弘 編著 ワークステーションシリーズ 統合化プログラミング環境 -Smalltalk-80 とInterlisp-D-，丸善，1987.