ソフトウェア工学 102-23
（1995. 1.27）

連結可能ウィンドウ・オブジェクト

喜瀬 浩　　新城 靖　　喜屋武 盛基
<kise@ocean.ie.u-ryukyu.ac.jp> <yas@ie.u-ryukyu.ac.jp> <kyan@ie.u-ryukyu.ac.jp>
琉球大学 情報工学科
〒903-01 沖縄県西原町千原1番地
Fax:098-895-2688

オブジェクトの堆積と連結は、object-basedシステムを構造化するためのモデルである。これらの特徴は、一様なインタフェースを持つオブジェクトを結合して、オブジェクトが持つ機能を統合して利用する点にある。本論文では、オブジェクトの連結モデルをウィンドウ・システムに適用した連結可能ウィンドウ・オブジェクトを提案している。そして、Xウィンドウ・システム上でウィジェット・レベルの連結可能ウィンドウ・オブジェクトのプロトタイプの実現方法を示している。実現方法の特徴は、パートナ・オブジェクトという概念が導入されたこと、およびウィジェットの手続き呼出しにRPC（遠隔手続き呼出し）を用いたことである。

Cascadable Window Objects

Hiroshi KISE　　Yasushi SHINJO　　Seiki KYAN
<kise@ocean.ie.u-ryukyu.ac.jp> <yas@ie.u-ryukyu.ac.jp> <kyan@ie.u-ryukyu.ac.jp>
Department of Information Engineering
University of Ryukyus
Nishihara, Okinawa 903-01, Japan
Fax: +81 98 895 2688

Object-stacking and object-cascading are models for structuring object-based systems. These models are used for integrating the functions of several objects which have uniform interfaces. This paper presents the implementation of cascadable window objects which are window objects based on the object-cascading model. This paper shows the implementation of prototypes of cascadable window objects at the widget level in the X window system. The features of the implementation are the introduction of the idea of partner objects and the use of RPC (Remote Procedure Call) for widget invocation.

1 はじめに

オブジェクトの堆積 (object-stacking) モデルは、object-based システムを構造化するためのモデルである [9]。このモデルでは、単純な機能を持つオブジェクトを組み合わせて、それらの持つ機能を合わせ持つようなオブジェクトを作り出すことができる。すでに、このモデルに基づく高機能ファイル・システムやインターネット上の情報提供システムが構築されている [6, 7, 9]。

ウィンドウ・システムにオブジェクトの堆積モデルを導入するために、そのモデルを拡張した**オブジェクトの連結 (object-cascading)** モデルを提案した [8]。これによって、ファイル・システムにおいて実現されたフィルタ処理のような高度な機能がウィンドウ・システムでも利用可能になる。また、ウィンドウ・システムの実現も容易になる。

オブジェクトの連結では、高度な機能を付加するために連結可能オブジェクトと呼ばれるものを用いる。本論文では、ウィンドウ・システムにおける連結可能オブジェクトである**連結可能ウィンドウ・オブジェクト (cascadable window objects)** の実現ついて述べる。本論文では、X ウィンドウ・システム上でウィジェット・レベルの連結可能ウィンドウ・オブジェクトのプロトタイプの実現方法を示している。実現方法の特徴は、パートナ・オブジェクトという概念が導入されたこと、およびウィジェットの手続き呼出しに RPC を用いたことである。

2 オブジェクトの堆積

オブジェクトの堆積モデルでは、一様なインタフェースを持つオブジェクトを積み重ねて（堆積させて）、それらの機能を合わせ持つオブジェクトを提供する。オブジェクトとは、データと手続きをカプセル化したもので、それぞれの固有の識別子、およびインタフェースを持つ。インタフェースとは、公開された手続きの集合である。インタフェースを一様にすることで、積み重ねるオブジェクトの構成と順序を自由に選択できる。オブジェクトは、サーバによって管理される。

下位層オブジェクトの上に上位層オブジェクトを**堆積させる**とは、上位層のオブジェクトが下位層のオブジェクトの識別子を持ち、上位層のオブジェクトが自分自身の機能の実現のために、下位層を呼び出すことである。

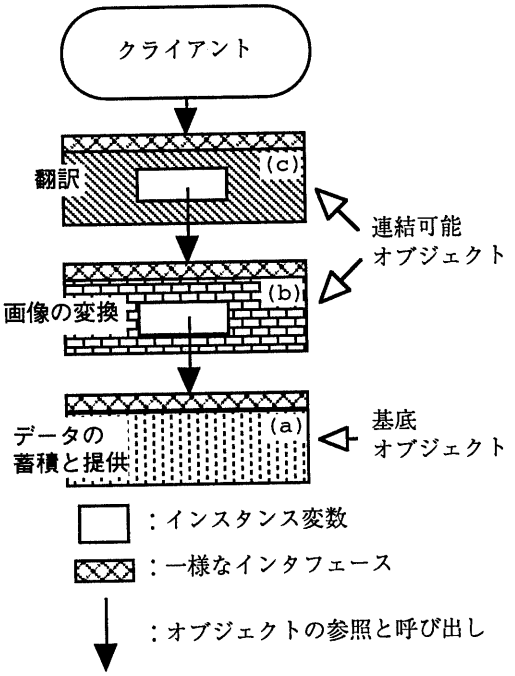

図 1: オブジェクトの堆積

図 1に、インターネットの情報提供システム W3[5] において 3 つのオブジェクトが堆積している様子を示す。上の 2 つのオブジェクトは、**堆積可能オブジェクト (stackable object)** で、内部に下位層のオブジェクトの識別子を保持している。最下位層のオブジェクトは、**基底オブジェクト (bottom object)** であり、ほかのオブジェクトのオブジェクト識別子を保持していない。

図 1では、次の 3 つの機能を持つオブジェクトが積み重ねられている。

(c) 翻訳

(b) 画像の解像度や色数の変換

(a) データの蓄積と提供

最上位層のオブジェクトをアクセスすることにより、これらの機能が統合される。画像部分が変換され、テキスト部が翻訳されたオブジェクトが作られる。

各堆積可能オブジェクトは、クライアント（上位層のオブジェクトのサーバを含む）から GET 要求

を受けると、対応する下位層のオブジェクトにクライアントとしてGET要求を送る。次に、下位層のオブジェクトから返されたデータに対し、翻訳あるいは画像の変換処理を行う。そして、結果をクライアントに返す。基底オブジェクトは、普通のW3サーバ内のオブジェクトで、他のオブジェクトを呼び出さず、ファイルに格納されているデータをクライアントに返す。

3 連結可能ウィンドウ・オブジェクト

3.1 オブジェクトの連結

ファイル・システムや前の章で述べたW3では、プロセス間通信におけるクライアント・サーバ・モデルが成り立っている。このモデルでは、プロセスが2種類に分類される。最初にメッセージを送り、それに対する応答を待つプロセスがクライアント、最初にメッセージが送られてくるのを待ち、それに対する応答を送るプロセスがサーバと呼ばれている。

ウィンドウ・システムにおいて、ディスプレイやキーボードを管理するものをカーネル、カーネルに働きかけてウィンドウを表示するものをアプリケーションと呼ぶことにする。ウィンドウ・システムの場合、プロセス間通信におけるクライアント・サーバ・モデルが成り立っていない。カーネルとアプリケーションの両方が能動的にメッセージを送るからである。このような場合、オブジェクトの堆積を利用することはできない。その理由は、下位層のオブジェクトは、上位層のオブジェクトの識別子を保持していず、そのため、上位層に対して能動的にメッセージを送れないからである。

この問題を解決するために、オブジェクトの堆積モデルを拡張する。オブジェクトに上位層かつ/または下位層のオブジェクトの識別子を保持させ、上位層や下位層に対して、能動的にメッセージを送ることができることにする。このモデルをオブジェクトの**連結モデル**と呼ぶ。このモデルでは、2種類のインタフェースを用いる。この一方を上方インタフェース、他方を下方インタフェースと呼ぶ。中間のオブジェクトは、上下に2つのインタフェースを持ち、上下それぞれのオブジェクトの識別子を保持する。このオブジェクトを**連結可能オブジェクト (cascadable object)** と呼ぶ（図2)。

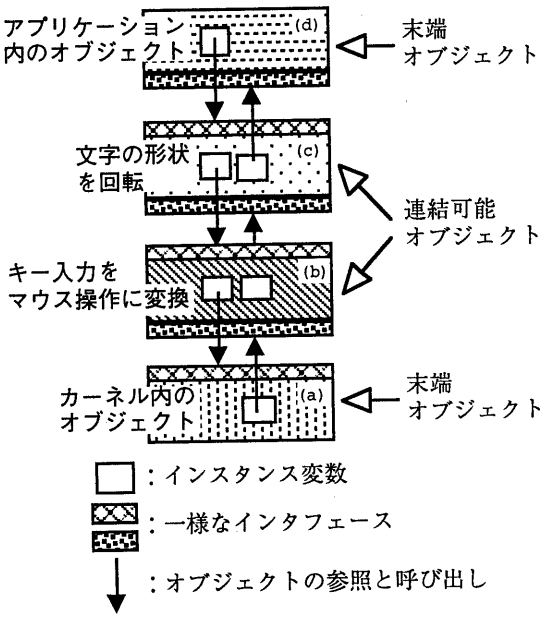

図 2: オブジェクトの連結

両端のオブジェクトは、一方向のみにインタフェースを持ち、上位層または下位層のオブジェクトの識別子を保持する。このオブジェクトを**末端オブジェクト (end object)** と呼ぶ。

本章では、連結可能オブジェクトを用いて、ウィンドウ・システムを構築する方法を述べる。

3.2 連結可能ウィンドウ・オブジェクトの動き

図2に、2つの連結可能オブジェクト、2つの末端オブジェクトが連結されている様子を示す。それぞれ、次のような機能を持つ。

(a) これは、末端オブジェクトであり、画面表示、入力の受付けを行う。

(b) これは、連結可能オブジェクトであり、画面に表示される文字の形状を回転させる機能を持つ。

(c) これは、連結可能オブジェクトであり、キーボードからの入力を、マウスによる操作に変

換する機能を持つ。例えば、左矢印キーを押すとマウスを左に動かしたことになる。

(d) これは、末端オブジェクトであり、イベントの受付けを行う。

オブジェクト (a) は、キー入力があると、メッセージをオブジェクト (b) に送る。オブジェクト (b) は、キー入力に対応するマウスの操作に置き換える処理を行い、オブジェクト (c) にメッセージを送る。オブジェクト (c) は、内容を変更しないでオブジェクト (d) にメッセージを送る。オブジェクト (d) は、マウス操作に対応した処理を行う。

アプリケーションが、文字を表示したい時、オブジェクト (c) に要求を送る。オブジェクト (c) は、文字の形状を回転させる処理を行い、オブジェクト (b) に文字表示の要求を送る。オブジェクト (b) は、内容を変更しないでオブジェクト (a) に要求を送る。オブジェクト (a) は、表示装置に文字を表示する。

3.3 連結可能ウィンドウ・オブジェクトの利用方法

連結可能ウィンドウ・オブジェクトによって可能になる機能には、以下のようなものがある。

- 部品の色やフォントを変更する。これは、X ウィンドウ・システムにおける資源の変更に相当する。
- ある表示装置に表示されている実行中のウィンドウ・アプリケーションを、別の表示装置に移動させる [1]。同じアプリケーションを複数の表示装置に表示する。
- キー入力やイベントのフィルタリングを行う。例えば、特定のボタンを無効化させたり、キーボードの操作によってマウスの代用をさせる。

我々のシステムでは、一様なインタフェースを要求しているため、プロトコルの拡張や変更ができない。X ウィンドウでは、X プロトコルと呼ばれる通信規約でカーネルとアプリケーションが通信している。X ウィンドウ・システムにおける LBX(Low Bandwidth X) は、X プロトコルのデータを圧縮して通信するプロトコルである。このようなプロトコルの変更は、一様なインタフェースに反する。

4 ウィジェット・レベルにおけるウィンドウ・オブジェクトの連結

ウィンドウ・システムでは、次のものがオブジェクトとみなせる。

1. ウィジェット **(widget)** をオブジェクトをみなすことができる。ウィジェットはウィンドウを構成する部品のことである。例えば、ボタン、メニュー、スクロールバーなどがある。
2. X プロトコル・レベルでは、ウィンドウ（画面に表示される矩形領域）、フォント（文字の形状）などがオブジェクトとみなせる [4]。
3. プロセスを 1 つのオブジェクトとみなすことができる。

本論文では、(2) の方法を用いる。この方法の利点としては、視覚的にオブジェクトと考えやすいこと、および手続きが明確に定義されていることがあげられる。

今回は、ウィジェットとして Athena Widgets を用いた。Athena Widgets は X ウィンドウ・システムの基本部分と共に配布されているため、入手が容易である。

4.1 ウィジェット・オブジェクトとパートナ・オブジェクト

Athena Widgets では、イベント処理をコールバックで実現している。XtAddCallback() によりコールバック手続きを登録する。イベントが発生すると、登録された手続きが呼び出される。その中で、発生したイベントに応じて処理を行う。

Athena Widgets に連結モデルを適用させるため、**パートナ・オブジェクト (partner object)** という考え方を導入する（図 3)。パートナ・オブジェクトは、ウィジェット・オブジェクトからのイベントを受け付ける。これは、Athena Widgets のコールバックに相当する。パートナ・オブジェクトは、ウィジェット・オブジェクトごとに 1 つずつ作られる。

パートナ・オブジェクトを導入する利点としては、呼出しが上下対称になることがあげられる。これに対し、Athena Widgets の場合は、アプリケーションからカーネルへの呼出しとカーナルからアプリケーションへの呼出しの方法がまったく異なる。

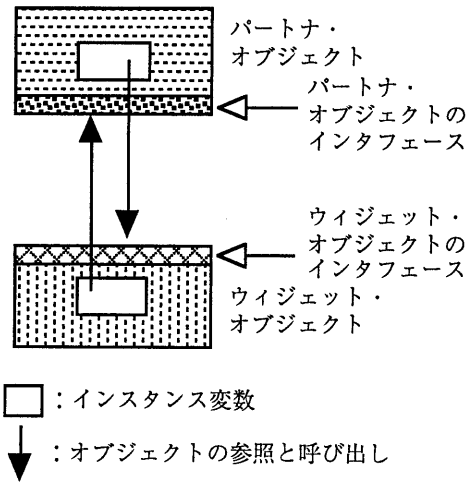

図 3: パートナ・オブジェクトとウィジェット・オブジェクト

ウィジェット・オブジェクトのインタフェースを以下に示す。

```
interface Widget {
  objectID createWidget(createArgs);
  void setPartner(objectID);
  void destroyWidget(void);
  resourceList getResource(resourceNames);
  void setResource(resourceList);
}
```

createWidget() は、ウィジェット・オブジェクトを生成し、その識別子を返す。引数は、ウィジェットの資源のデータである。setPartner() は、上位層のオブジェクトへの参照を形成する。引数は上位層オブジェクトのの識別子である。destroyWidget() は、このオブジェクトを破壊する。

getResource() は、ウィジェットの資源の値を調べる。引数は、調べる資源の名前のリストである。返値は、資源の名前と資源の値の対からなるリストである。setResource() は、資源の値の設定をする。引数は、資源の名前と資源の値の対からなるリストである。それぞれの Athena Widgets に対応する手続きは、XtGetValues() とXtSetValues() である。

パートナ・オブジェクトのインタフェースを以下に示す。

```
interface Partner {
  objectID createPartnet(createArgs);
  void setWidget(objectID);
  void destroyPartner(void);
  void event(eventType, eventData);
}
```

createPartner() は、パートナ・オブジェクトを生成し、その識別子を返す。引数は、ウィジェットの資源のデータである。setWidget() は、下位層のオブジェクトへの参照を形成する。引数は、下位層オブジェクトの識別子である。destroyPartner() は、このオブジェクトを破壊する。

event() は、イベントの発生を通知する。引数は順に、イベントの種類、ウィジェットから渡されたイベント固有のデータである。例えば、スクロールバー・ウィジェットの場合、イベントの種類は、マウス・ポインタの位置の変更、イベント固有のデータは、バーの現在の位置である。

4.2 連結可能ウィンドウ・オブジェクトの実現方法

連結可能ウィンドウ・オブジェクトは、間接オブジェクト [7] と同様に実現される。連結可能ウィンドウ・オブジェクトのデータとしては、上下層のオブジェクトの識別子やインスタンス変数がある。連結可能ウィンドウ・オブジェクトの手続きの手順を以下に示す。

1. 受け付けたメッセージ中のオブジェクト識別子を内部の構造体へのポインタに変換する。
2. 対応する上位層または下位層オブジェクトへ自分自身と同じ名前の手続きを呼び出す。その際、引数の値を加工する。オブジェクトの識別子を、上位層または下位層オブジェクトの識別子に置き換える。
3. 返値を加工する。

4.3 Athena Widgets の RPC 化

オブジェクトの連結モデルでは、同じ機能を持つオブジェクトは、1 つのサーバにより管理させる。オブジェクト間の通信は、サーバ間のプロセス間通信となる。今回は、これを RPC(Remote Procedure Call) で実現した。

Athena Widgets では、ウィジェット操作をする手続きの引数に構造体へのポインタを用いる。今回は、

構造体のポインタの32ビットのイメージをそのままプロセス外のオブジェクト識別子とした。これが可能である理由は、Athena Widgetsを利用するプログラムが、構造体のフィールドを直接操作することがないためである。今回は、ポインタをそのままオブジェクトの識別子として扱ったが、本来は偽造や改変に備えて、チェックしなければならない。Athena Widgetsでは、ウィジェット・クラスを表現するため、静的な構造体のポインタを使っている。このポインタは、パートナ・オブジェクト側のプロセスの初期化時にRPCによって渡される。

オブジェクト間の通信にSunRPCを用いている。SunRPCの仕様から、入力パラメタと出力パラメタを1つの構造体にまとめる必要がある。Athena Widgetsの手続きには、入出力パラメタがあるが、SunRPCでは扱えないので、出力パラメタに含むようにした。

Xウィンドウ・システムのカーネルに我々の提案した上方インタフェースを組み合わせたものを、本システムのカーネルとみなす。連結可能ウィンドウ・オブジェクトを管理するサーバの内部における処理を並列に実行するため、軽量プロセスを用いている。

5 関連研究

Springシステムは、分散ファイル・システムやキャッシュの整合性を実現するファイル・システムを持っている[3]。その実現において、1つのサーバで、互いに関連した2種類のオブジェクトを提供する。ここで、互いに関連した2種類のオブジェクトを、2つのインタフェースを持つ1つのオブジェクトと考えると、我々の言葉でいう連結可能オブジェクトに相当する。

文献[2]では、オブジェクトの概念のないインタフェース(flat interface)にinterposing agentと呼ばれる層をアプリケーションとカーネル間にはさみこむことで、XやUNIXの機能を拡張する方法について述べている。この論文は、interposing agentの実現を支援するために、内部でオブジェクト指向を取り入れる方法を示している。我々の提案した方法は、外部でオブジェクトを用いているシステムに対して利用できる。

6 まとめ

本論文では、オブジェクトの連結に基づく連結可能ウィンドウ・オブジェクトの実現について述べた。本論文で提案した方法によって、高度な機能を持つ連結可能ウィンドウ・オブジェクトが提供される。今回は、ウィジェットのレベルで、オブジェクトの連結を実現する方法を示した。その方法の特徴は、パートナ・オブジェクトの概念を導入したこと、および手続き呼出しにRPCを用いたことである。

今後は、連結可能ウィンドウ・オブジェクトに適したインタフェースを決定し、ウィンドウ・システムを構築したい。

参考文献

[1] 一柳, 北川: "Xウィンドウ・マイグレーション機構とその一実現例", 情報処理学会第47回全国大会講演論文集 (5), 2D-2, pp. 145–146 (1993).

[2] M.B.Jones: "Using Objects to Build Derived Implementations of Flat Interfaces", IEEE Proc. of 2nd International Workshop on Object Orientation in Operating Systems (IWOOOS'92), pp.341–345 (1992).

[3] Y.Khalidi and M.Nelson: "Extensible File Systems in Spring", 13thSOSP, ACM Operating System Review, Vol.27, No.5, pp.1–14 (1993).

[4] A.Nye: "X Protocol Reference Manual", O'Reilly & Associates, Inc. (1990)

[5] T.Berners-Lee, et.al: "The World-Wide Web", Communications of the ACM, Vol.37, No.8, pp.76–82 (1994).

[6] 新城: "オブジェクトの堆積・連結モデル", 情報処理学会第6回コンピュータ・システム・シンポジウム, Vol.94, No.10, pp.31–38 (1994).

[7] 島袋, 新城, 翁長: "オブジェクトの堆積モデルに基づく間接オブジェクトの実現", SWoPP '94, 情報処理学会研究会報告, 94-OS-65-15, Vol.94, No.64, pp.113–120 (1994).

[8] 新城, 武川, 翁長: "オブジェクトの堆積・連結モデルに基づくウィンドウ・システム", 情報処理学会第48回全国大会講演論文集 (5), 2K-8, pp.(5)341–342 (1994).

[9] Y.Shinjo and Y.Kiyoki: "The Object-Stacking Model for Structuring Object-Based Systems", IEEE Proc. of 2nd International Workshop on Object Orientation in Operating Systems (IWOOOS'92), pp.328–340 (1992).