オペレーティング・システム　52－2
（１９９１．　９．　６）

Window Shared Data Pool

越塚 登　高田 広章　坂村 健

東京大学理学部情報科学科

本稿はGUIプログラミングの労力の軽減を目的とした、BTRONの新しいウィンドウシステムについて述べる。そのアーキテクチャやプログラミングインタフェースは、グループウェアや、カット＆ペースト機能、付箋を使ったバッチ処理メタファなどのような、新しい対話スタイルのGUIのプログラミングがサポートできるように設計されている。

このGUIプログラミングモデルのベースとして、本稿では始めに対話モデルを提案する。そして、その対話モデルに基づいた、プログラミングインタフェースモデルである、Abstract Event/Window Callセットを提案する。そして、この、Abstract Event/Window Callセットを実際に実現するソフトウェアである、Window Shared Data Pool (WSDP) についても述べる。

また、本稿ではWSDPのプロトタイプバージョン上に、このプログラミングインタフェースを生かして、分散共有黒システムを少ないソースコード量で実装できたことも加えて報告する。

Window Shared Data Pool

Noboru Koshizuka　Hiroaki Takada　　Ken Sakamura

Department of Information Science, Faculty of Science, University of Tokyo

7-3-1, Hongo, Bunkyo-Ku, Tokyo 113, Japan

This paper describes a new window system for BTRON which aims to decrease the effort of GUI programming. It is characterized by its architecture and programming interface, which can support the programming of groupware-like style of GUI, cut & paste functionality, "automatic-execution of fusen", etc.

As a basis of our GUI programming model, this paper first proposes a dialogue model, and, next, proposes *Abstract Event/Window Call* set: a GUI programming interface based on the dialogue model. It also proposes *Window Shared Data Pool* (WSDP): software which realizes the Abstract Event/Window Call set.

This paper also reports that we have constructed a distributed shared blackboard program on its prototype with small amount of source code.

1 はじめに

近い将来、コンピュータ技術の進歩により、コンピュータは生活のあらゆる場面で使用されるようになることが予想される。コンピュータのユーザ層もそれに伴い拡大しなけらばならない。これは、コンピュータの非専門化のユーザという意味だけでなく、ユーザの持つ障害や、読み書きできる言語などの様々な個人の特質を問わずコンピュータを使用できるという意味を持つ。よって、TRONアーキテクチャのコンピュータは、このような広い応用とユーザ層に対しユーザフレンドリなヒューマンマシンインタフェース (HMI) を提供することを目指している [11]。そのためにTRON プロジェクトでは、EnableWare[12] (障害者仕様コンピュータ)、多国語コンピュータ [14]、Solid/Special User Interface (SUI: 機器組み込みコンピュータ用HMI)、電子文房具 など、幅広くHMIの研究を進めているが、実現にはまだ多くの新しい技法の開発が必要である。

グラフィカルユーザインタフェース (GUI) はワークステーション上のユーザフレンドリなHMIとして、欠かすことの出来ないHMI技法の一つである。先程述べた様に、HMIの分野は課題を多く抱えているが、現在のGUIだけでも問題は山積しており、本稿では、こうした幅広いHMIの課題の中からTRONプロジェクトにおけるGUIシステムへの取り組みを述べる。

GUIには提供する機能に関する問題もあるが、GUIの最大の問題点の一つに、そのプログラミングの難しさがある。GUI実現のための複雑なデータ構造やアルゴリズムのプログラミングの結果、アプリケーションソフトウェアに占めるGUI部分のコード量が多大になり、中には約40% ～ 50% にも達するという報告さえある [1]。これが、GUIを提供するアプリケーションプログラムの効率的な開発を妨げる最も大きな原因となっている。

こうした問題を解決するために、GUIプログラミングをサポートする研究が行なわれてきた。ウィンドウシステムは、GUIプログラミング、実行をサポートする、現在最も一般的なソフトウェアシステムである。また、より高度なレベルでGUIプログラミングをサポートするシステムとして種々のツールキットやUser Interface Management Systems (UIMSs)[3] が現在まで盛んに研究されている。

ところが、近年のコンピュータの用途、応用の拡大によって、それらが想定していなかったGUIシステムに対する要求が新たに加わってきた。例えば、

1. 異なるアプリケーション間でのカット & ペースト操作によるデータ交換、
2. グループウェアやCSCWで用いられる、複数のユーザ間のウィンドウ共有機能 [2] や、
3. 複数アプリケーションプログラムの協調処理機能を提供することで、アプリケーションソフトウェアの *Down Sizing* を促進すること

などが挙げられる。

GUIシステムの殆んどは、人間とコンピュータの間の“対話モデル”に基づいて設計されているが、従来のものは、対話モデルのレベルでこうした新しい対話スタイルを全く考慮していなかった。そのため、従来のGUIシステムでは、これらの、新しい対話スタイルのプログラミングをサポートすることができない。よってこの要求に答えるためには、まず、GUIのシステムの基盤となる、“対話モデル”の再構築、次にその対話モデルに基づいたGUIシステムの設計が必要である。

そこで本稿では、まず始めにこのような新しいGUIの対話スタイルも包含し、GUIのモデル化に必要な要求を満たすHMIモデル「マグネットボードモデル」を提案する。次に、マグネットボードモデルに従い、新しいGUI応用のプログラミングにも適した、新しいBTRONウィンドウシステムと、そのコアである共有メモリメカニズム (Window Shared Data Pool: WSDP [4, 5]) を提案する。そのアーキテクチャは、分散共有メモリをシステムのコアに置き、先の1を解決するために、異なるアプリケーションプログラム間でのデータ交換機能のプログラミングのサポート、そして2の機能実現のために、複数のユーザ間での同じビューを持ったウィンドウ（分散共有ウィンドウ）のプログラミングサポートを行なう。また3の機能を実現するために異なるアプリケーション間でのウィンドウデータ共有のプログラミングのサポートを行なう。

2 対話モデル

2.1 対話モデルへの要求

対話モデルとは人間とコンピュータシステムの間の対話形式を自然に表現することが目的であり、今回特にその表現力に重点を置いて対話モデルを構築した。

ここでは、まずグラフィクスをメディアとしたGUIによる対話を自然に表現することが必要とされる。そして対話の主体である人間とコンピュータの間の、以下のような様々な対話スタイルを表現できなければならない。まず、通常のアプリケーションにおけるHMIと同じように、一人のユーザが一つのアプリケーションプログラムと対話するスタイル（1:1対話）である。

また、グループウェアやCSCWように、複数のユーザが単一のアプリケーションと対話するスタイル（N:1

対話）も表現できなければならない。

加えて、複数のアプリケーションをユーザが協調させて一つのタスクをこなすような対話スタイル（1:N対話）がある。この協調処理の実現には、複数アプリケーション間でのデータ共有が不可欠となるが、データ共有の方法には二通りある。

一つは異なるアプリケーションウィンドウ間のカット＆ペースト操作で、必要な部分データだけを移動する方法である。

もう一つは、処理データ全体を複数のアプリケーションで共有する機能である。例えば、文書の編集、スペルチェック、印刷処理、単語のサーチ処理などを別々のアプリケーション[1]として用意し、これらが単一の文書ウィンドウのデータを共有して協調処理するものである。現在のGUIシステム上にはこの機能がないために、アプリケーションソフトウェアは自己完結の巨大なものとなり、その開発、保守に問題が生じる。我々は、1:N対話スタイルをウィンドウシステムがサポートすることで、アプリケーションの巨大化を防ぎ、アプリケーションの *down sizing* を促進し、その開発、保守を容易にすることを狙っている。

従来、UIMSの研究の中で多くの対話モデルが提案されていが、ここで挙げたような要求をすべて満たすような対話モデルは存在しない。殆んどの対話モデルがこのN:1対話や、1:N対話などの人間とコンピュータの新しい対話スタイルを表現することができない。

2.2 マグネットボードモデル

従来の対話モデルは、新しい対話スタイルの自然な表現には不十分であった。よって本節では新しい対話モデル「マグネットボードモデル」を提案し、その概略を述べる。まず大きくHMIをユーザとアプリケーションプロセス間の通信とモデル化する。そして、HMIで用いられるウィンドウとは、その通信のためにユーザとアプリケーションプログラム間で共有される資源で、その中に複数の対話用のオブジェクトを含む。

この対話モデルは以下の四種類の要素から構成されている（図1）。

ウィンドウオブジェクト はウィンドウ内に存在する、ユーザ、アプリケーションプログラム間の通信に利用される対話オブジェクトである。これらは、ウィンドウ内では個々に並列／独立に動作する。

ウィンドウ は、その中にウィンドウオブジェクトを含み、複数のユーザ、アプリケーションプログラムから共有される。ウィンドウ外部からは、ウィンドウオブジェクトを 1) read, 2) set, 3) update, 4) operate, 5) delete という操作を行なえる。そのウィンドウを共有しているユーザ、アプリケーションプロセスは、その操作を即座に知ることができる。

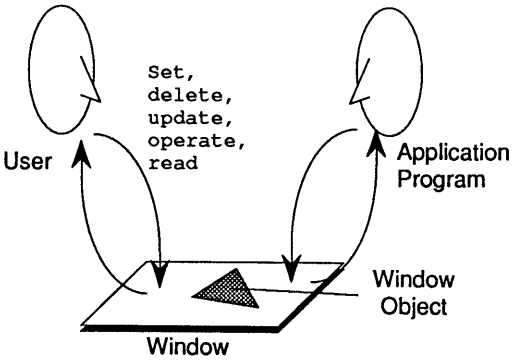

図1: マグネットボードモデル

ユーザ、アプリケーションプログラム はウィンドウを共有する要素で、このモデルでは対等な要素とみなされる。これらの要素は、ウィンドウに対し、先の五つの操作を互いに非同期に加えることができる。

メタファーをとれば、ウィンドウとは、ユーザ-アプリケーションプログラム間の対話に使う、マグネットボード（Magnetboard）のようなものある。そこに、通信に用いる図形、インタラクションテクニックなどのactiveなオブジェクトが、磁石でつけられる。ユーザもアプリケーションもこれらをつけたり、はがしたり、置き換えたりなど、そのオブジェクト操作することで通信する。このメタファーから、ここで提案する対話モデルを「マグネットボードモデル」と呼ぶ。

マグネットボードモデルは、先に挙げたような様々な対話スタイルを自然に表現することができる。まず、N:1対話スタイルとして代表的な分散共有黒板のようなグループウェアは、ウィンドウを、一つのアプリケーションプログラムと複数のエンドユーザが共有するという、マグネットボードモデルの単純な変形で表現できる（図2右上）。

一方、1:N対話で必要とされる、異なるアプリケーションプログラム間のカット＆ペースト機能は、左から右のウィンドウへユーザがウィンドウオブジェクトを剝して移し替えると表現する（図2左上）。また複数アプリケーションによるデータ共有も図2右上図のユーザ側とアプリケーションプログラム側の立場を逆にして、複数アプリケーションプログラムが単一のウィ

[1] この協調処理を前提として、個々の用途に特化して開発されたアプリケーションを我々は特に、「文具ウェア」と呼んでいる。

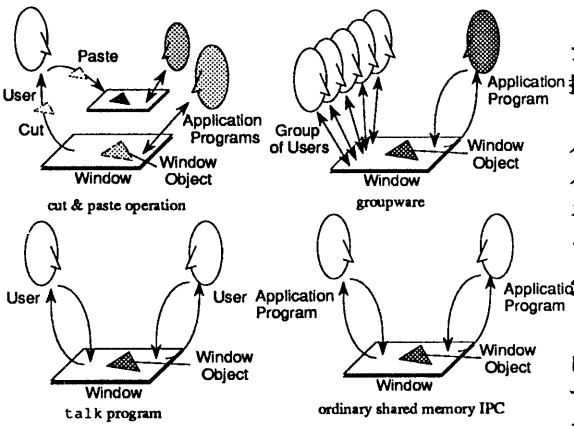

図 2: マグネットボードモデルのバリエーション

ンドウを共有する形へ変更すれば簡単に表現できる。

加えて、マグネットボードモデルは一般性の高い通信モデルで、HMI のみならずアプリケーションプログラム間通信や、talk プログラムなどのユーザ間の通信なども統合的に表現できる（図 2下）。

3 BTRON ウィンドウシステムのアーキテクチャ

本節では、第 2 節で述べたマグネットボードモデルに基づいた新しい BTRON ウィンドウシステムについて述べる。

3.1 BTRON

ここで、まずマグネットボードモデルをマップする対象である BTRON についての概略と、BTRON プロジェクトにおける本研究の位置付けについて簡単に述べる。

BTRON とは TRON プロジェクトにおける HMI を担当するコンピュータである [6]。BTRON では、アプリケーションプログラム間のデータ互換 (TAD 仕様 [10]) と、HMI 作法の標準化 (BTRON HMI 仕様 [10]) が最も重要視されている。従来これを実現するカーネル仕様として、パーソナルコンピュータ用の BTRON1[10] と、TRON VLSI CPU を用いたワークステーション用の BTRON2[8] がある。これらの上には同じウィンドウシステムがカーネルベースで実装されている。本研究は、BTRON HMI 仕様を実現する、BTRON カーネル上の新しいウィンドウシステムの提案と位置付けることができる。

データと操作の互換性以外にも、BTRON は、ポップアップメニューや、インタラクションテクニックを提供するパーツを使った、マルチタスクマルチウィンドウという操作環境、GUI の実現に必要なソフトリアルタイム性、ハイパテキスト構造の実身／仮身ファイルシステム、障害者のアクセスのサポート、シームレスな多国語処理、電子文房具との接続などの特長がある。

3.2 設計方針

BTRON HMI 仕様を実装するウィンドウシステムに対して、次のようにいくつかのより具体的なレベルでの設計方針を持つ。まず、1) マグネットボードモデルに基き、2) 分散環境に対応、3) 異なるアプリケーション間でデータ交換を行えること、4) 効率的な GUI プログラミングが可能なことがある。この効率的な GUI プログラミングを実現するためには、高レベルでシンプルなウィンドウシステムインタフェースを提供することが必要とされる。また、5) ソフトウェアアーキテクチャのモジュールの対称性も持たせるようにする。

3.3 BTRON ウィンドウシステムのアーキテクチャの概観

マグネットボードモデルを BTRON システム上にマップすると、BTRON ウィンドウシステムのソフトウェアアーキテクチャは、そのモデルの各構成要素に対応した以下の構成要素から成り立つ。

ウィンドウオブジェクト は BTRON HMI 仕様で定められた、ウィンドウ内に表示され、ユーザとコンピュータの対話に用いられるグラフィカルオブジェクトである。以下で述べる WSDP に格納されるデータの基本単位となる。

Window Shared Data Pool (WSDP) はウィンドウを実現する分散共有メモリである。ウィンドウ一つに対して一つ存在し、そのウィンドウを通して、ユーザまたはアプリケーションと通信するプロセスは、WSDP を共有する必要がある。その WSDP は次のような役割を持つ。

アクティブなウィンドウオブジェクトをデータの基本要素として格納し、外部からのアクセスインタフェースとして、マグネットボードモデルに基づいたアクセスインタフェース（*Window Call*[13][2]）、アクセス対象のウィンドウオブジェクトを指定するための複数種類のアドレスシステム（マルチアドレシングシステム）、WSDP に対する操作通知するイベントメッセー

[2][13] では、*IM* コールと呼んでいる。

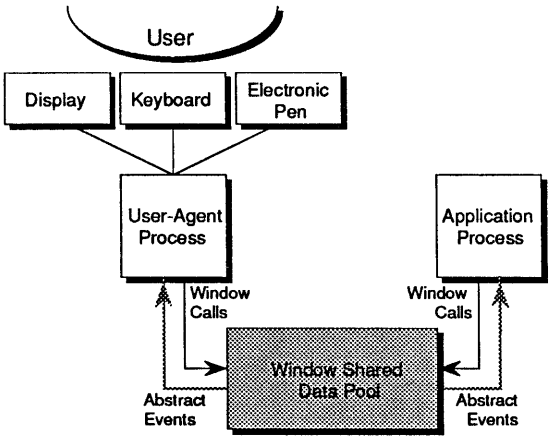

図3: マグネットボードモデルに基づいた、新しいBTRONウィンドウシステムのアーキテクチャ

ジ (*Abstract Event*[13]) などの特色を持つ、詳しくは第4節で議論する。

ユーザエージェントプロセス は、ユーザエージェントプロセスはマグネットボードモデルでは、エンドユーザに相当するプロセス。ディスプレイに対して一つ存在し、コンピュータの中で、ユーザの代理人として動作する。

具体的には、ディスプレイ資源を各ウィンドウ用に分割、割当、そのウィンドウ内部へ物理的な描画処理、描画に必要な情報のWSDPからの取得、デバイス独立なインタフェース提供や、入力の各ウィンドウへの分配、入力位置データの各ウィンドウ局所座標系への変換、入力に従ったWindow Callの発効などを行なう。

アプリケーションプロセス は、マグネットボードモデルにおけるアプリケーションプログラムに対応し、アプリケーションプログラム毎に存在する。エンドユーザからの入力が必要な時は、Abstract Eventによって非同期に通知されるのを待つかまたは、WSDPから、陽にウィンドウオブジェクトデータをreadする。逆に出力する時は、ウィンドウオブジェクトをWSDPにsetする。

これらの要素の関係を図3に示す。

4 Window Shared Data Pool

BTRONウィンドウのアーキテクチャモデルでは、アプリケーションプロセスがWSDPと通じてGUIを実現する。よって、WSDPのアクセスインタフェースや構造が、GUIのプログラミング効率や、GUIの性能を大きく左右する。分散共有メモリにおいて重要な特性項目には、共有メモリ中に格納される記憶要素のデータモデル、そのデータに対する操作モデル、そのデータへのアクセス方法（アドレッシング）がある。本節では、WSDPについてこれらの項目をそれぞれ詳細に述べる。

4.1 セマンティクス

WSDPはAdvanced Shared Memory[9]をBTRONウィンドウ応用に特化した分散共有メモリの一つである。よってWSDPはそのデータモデル、操作モデル、アドレッシングと、すべてBTRONウィンドウの実現に適するように設計されている。

WSDPは単に静的なデータをset, delete, updateするだけの共有記憶ではない。そこには、reactiveな記憶要素（ウィンドウオブジェクト）が格納され、外部からの操作によって、WSDP内部でそれらは動作することができる。WSDPは、そうした動作を行なう場と、外部からの操作アクセスをサポートする構造を与えている。

WSDPはウィンドウのスクロール、リサイズによって、物理的なウィンドウ上に表示される可能性のある、ウィンドウオブジェクトを含んだ全2.5次元状[3]の空間を指す。図形エディタでいえば、WSDPは、画面に物理的に見えている図形データだけでなく、現在ウィンドウの外に位置しスクリーン上には見えないが、図形エディタの編集対象となっている図形データをも含む。

マグネットボードモデルのように、WSDPを論理的に単一モジュールと捉えることが出来るが、分散環境にWSDP実装した場合などは、物理的にはレプリカがマシン間に分散して存在する実装も可能である。

4.2 データモデル

BTRONウィンドウには、インタラクションテクニックや対話的処理ができるデータとして用いられるウィンドウオブジェクトが置かれる。ウィンドウオブジェクトとは、1) 図形／文章、2) 仮身、3) コントロールパーツ、4) メニューの四種類のグラフィカルなオブジェクトをまとめていう。このウィンドウオブジェクトの種類、形状、それらのウィンドウ上での動作などは、BTRON HMI仕様で定められている。

WSDPにも、このウィンドウオブジェクトが格納されるわけだが、その物理的なデータ形式もアプリケーション独立に標準化されている。その標準の枠組がTAD (TRON Application Data-bus)[10]であり、個々のウィ

[3] しばしばウィンドウは、x-y軸の2次元と、順序関係だけのz軸を0.5次元と考え、2.5次元空間と呼ばれる。

ンドウオブジェクトをパラメータ表現したデータ単位をTADセグメントと呼ぶ。具体的なデータ構造は[10]に示されている。

WSDPに格納される基本データ要素もこのTADセグメントであり、かつカット&ペーストなどによるデータ交換、実身[4]でのデータ蓄積の基本フォーマットでもある。このようなデータモデルの標準化により、異なる複数のアプリケーションプログラム間のデータ共有が可能になる。

4.3 操作モデル

WSDPに対する操作モデルは、マグネットボードモデルに基づき、ウィンドウオブジェクトを1) `Set`, 2) `Delete`, 3) `Update`, 4) `Operate`, 5) `Read`するという操作を基本としている。そしてこれに実装上、複数ユーザ間または、ユーザアプリケーション間の同期をとるために、これにオブジェクトを6) `Lock/Unlock`するというコールと、後で述べるAbstract Eventを要求するコール7) `GetAbstractEvent()`が加えられる。この7種類の基本コールを*Window Call*セットとよび、WSDPに対してユーザエージェントプロセスからも、アプリケーションプロセスからも同様にこのコールを発行することが出来る。

Window Callsの中で、操作対象であるウィンドウオブジェクトの指定は、WSDPの提供する高機能なマルチアドレシング機能（4.4節参照）でのアドレスを用いる。

ウィンドウの描画環境などのコンテキストの変更、設定、ウィンドウ自体へのアクセスインタフェースなどは厳密には、Window Callではない。しかし、これらのコールはすべてウィンドウプログラミングに必ず必要なものなので、同じ関数名形式に統一してこれらのアクセスインタフェースを提供した。

ウィンドウを純粋な共有メモリ的な資源として実現する場合の問題点は、ウィンドウの変更を、そのウィンドウを共有するアプリケーションプロセスや、ユーザエージェントプロセスが効率良く知るメカニズムが存在しないことである。常にWSDP全体をポーリングする実装は性能上現実的でない。そこで、WSDPは各共有プロセスに対し、WSDPへの他プロセスからの操作を通知するイベントメカニズム、*Abstract Event*、を持っている。

Abstract Eventは、「どのような型（ウィンドウオブジェクトの型を指定）のどのオブジェクト（idを指定）が、どういう操作（Window Callの型を指定）をされたか」を通知するメッセージである。

[4]BTRONの実身／仮身ファイルシステムモデルにおけるファイル。

WSDPが外部から副作用のあるアクセスをされると、Abstract Eventを、WSDPを共有しているプロセスに送る。よってAbstract Eventも、Window Callと同様の、1) `Set` 2) `Delete` 3) `Update` 4) `Operate` 5) `Lock/Unlock`というイベントタイプがある。

アプリケーションプログラムは、Abstract Eventによるイベントループを構成する形式で記述される（付録A参照）。アブストラクトイベントは、従来のウィンドウシステムの、ウィンドウイベントのように、イベント発生位置と、入力デバイスの状態を知らせるものよりも、はるかに抽象度が高い情報を通知している点で全くことなる。

4.4 マルチアドレシングシステム

記憶空間では、その中のデータにアドレスがつけられており、それを利用してある数学的集合上のキーからデータへアクセスする写像、アドレッシング、が必要とされる。WSDPの分散共有メモリとしての大きな特徴は、アドレッシングを複数パターンサポートしており、それぞれのアドレッシングが、GUIプログラミングにおいて、ウィンドウオブジェクトの参照パターンに適する様に設計されていることである。このアドレッシングシステムをマルチアドレシングシステムと呼ぶ。

CPUの設計においても、適切な高機能のアドレッシングを提供することで、プログラミングを軽減することができる。これと同様、WSDPもGUIプログラミングに適した高機能なアドレッシングシステムを提供することで、プログラミング労力の軽減を行なう[9]。

WSDPでは、図4にあるようにウィンドウオブジェクトに対して、GUIプログラミング時に有用な、以下の二つのアドレスを与えることが出来る。一つは、1) ウィンドウオブジェクトのId番号を直接アドレスとするもので、もう一つが、2) そのウィンドウオブジェクトがウィンドウ上で占める2.5次元中の領域をアドレスとするものである。このアドレスを基盤として以下の三つの高機能なアドレッシングを提供している。

1. Id番号 $\mapsto$ そのId番号をアドレスとするウィンドウオブジェクト

   これは、「id = 245番のオブジェクトに対して、`delete`操作を加える」というようなアクセスに用いられる（図4上のアクセス）。

2. ウィンドウ上の点 $\mapsto$ その点の直下のウィンドウオブジェクト

   「ウィンドウ上の点(x, y)直下のウィンドウオブジェクトを`operate`する」というアクセスが可能（図4下のアクセス）。

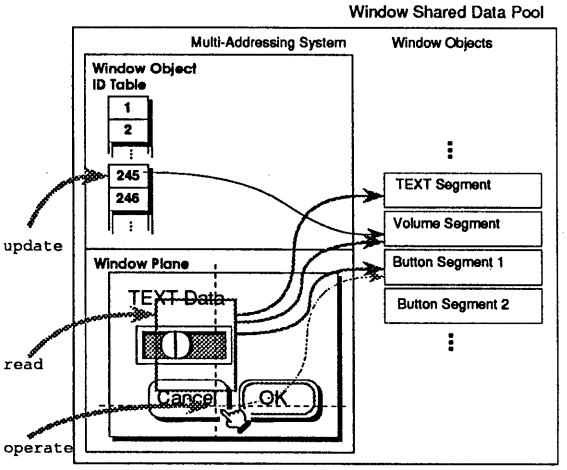

図 4: マルチアドレシングシステム

3. ウィンドウ上の領域 ↦ その領域と重なりを持つウィンドウオブジェクト列

   これにより、「ウィンドウ上の指定された領域と重なりのあるウィンドウオブジェクト群に、read という操作を加える」というアクセスが可能になる（図4真中のアクセス）。

5 WSDP のプロトタイプの実装と、プログラミング例

今回、BTRON1 の既存のウィンドウシステム上にツールキット形式で、WSDP のうち、Window Call インタフェースと Abstract Event 発生機構、ウィンドウオブジェクトの TAD 標準フォーマットサポート部分のプロトタイプシステムの実装を行なった。その上で、WSDP の外部インタフェースを用いた、プログラミング効率の評価と、マグネットボードモデルの有効性を検証するために、簡単な分散共有黒板システムを作成した。

分散共有黒板システムとは、複数ユーザのディスプレイ上に、同じビューを持ったウィンドウが表示され、その複数のユーザは、自由にそのウィンドウを操作でき、その操作は全員のウィンドウ中に反映されるというものである。この分散共有黒板は、簡単なテキストと、TAD 標準仕様でサポートされている図形を扱うことや、それらを他のローカルウィンドウからデータのカット & ペーストを行なうといった機能を持たせた。

WSDP 上ではこの分散共有黒板のようなプログラミングを特に効率良く行なうことが出来るが、その理由には三つある。

1. 黒板ウィンドウの共有機能は、WSDP が分散共有記憶として実装されているので、アプリケーションプログラマがプログラムする必要はない。ウィンドウ作成時に共有可能属性を設定し、複数のユーザエージェントが同一の WSDP を共有すれば、この共有機能はプログラマ透明に実現することができる。

2. その黒板へのカットやペースト操作の処理は、アプリケーションプログラマが記述しなくとも、WSDP 側で自動的に行なう。アプリケーションプログラムは、その結果としての Delete や Set イベントを受けとるだけである。

3. その他の部分に関しても WSDP は高抽象度のプログラマインタフェースを使うことが出来る。

この分散共有黒板を WSDP のプロトタイプ上に C 言語で約 110 行程度（コメント、空行も含む）で実装することが出来た（その一部は付録 A に示してある。）。

6 おわりに

本研究では、通常の GUI のプログラミングに加えて、今までウィンドウシステムによってプログラミングサポートが行なわれていなかった、グループウェアのようなウィンドウ共有機能や、複数のアプリケーションプログラム間のデータ共有のプログラミングも簡単にする、マグネットボードモデルに基づく、新しい BTRON ウィンドウシステムを提案した。そのために、本システムは、高レベルな Abstract Event/Window Call セットプログラミングインタフェースと、ウィンドウオブジェクトアクセスをサポートするマルチアドレシングシステムを提供する。WSDP はそれ以外にも、三次元ウィンドウへ容易に拡張可能であること、疎結合分散環境に実装した際に、ネットワーク通信データ量が少ないなど様々な利点がある [4]。

本稿では触れられていないが、WSDP の枠組は GUI 分野だけにとどまらず、様々なメディアを使い、ユーザの特性[5]に適応した HMI を提供する HMI システムのプラットホームへ発展することができる。

そこでは、アプリケーション側からは WSDP に、ウィンドウオブジェクトのような具体的なインタラクションテクニックのインスタンスでなく、HMI の要求仕様の抽象表現が与えられる。それらは WSDP の中でユーザの特性に合わせて個々の具体的なインタラクションテクニックとして動作する。この抽象表現は

[5]目が見えない、上肢の動作が不自由、日本語しか使えないなど。

TACL (TRON Application Control-flow Language)[7]という言語で記述され、WSDPはTACLプログラムを評価／実行することのできるインタプリタとしての性質を持つことになる[4]。この機能によって、障害者、様々な国籍の人々を含めた多くのユーザに対して、彼らの特性合わせたにHMIを自動適応させることができる。

参考文献

[1] D. G. Borrow, S. Mittal, and M. J. Stefik. Expert Systems: Perils and Promise. *Communications of the ACM*, Vol. 29, No. 9, pp. 880–894, September 1986.

[2] C. A. Ellis, S. J. Gibbs, and G. L. Rein. Groupware: Some Issues and Experiences. *Communications of the ACM*, Vol. 34, No. 1, pp. 38–58, January 1991.

[3] H. R. Hartson and D. Hix. Human-Computer Interface Development: Concepts and Systems for its Management. *ACM Computing Surveys*, Vol. 21, No. 1, pp. 5–92, 1989.

[4] N. Koshizuka. A Distributed Shared Memory Model for Window System Construction and its Extension to a General Shared Memory Framework. Master's thesis, Division of Science, Graduate School of University of Tokyo, February 1991.

[5] N. Koshizuka, H. Takada, and K. Sakamura. Window Shared Data Pool: A New BTRON Window System. in preparation.

[6] K. Sakamura. BTRON: The Business-Oriented Operating System. *IEEE MICRO*, Vol. 7, No. 2, pp. 53–65, April 1987.

[7] K. Sakamura. TACL : TRON Application Control-flow Language. In Ken Sakamura, editor, *TRON Project 1988*, pp. 79–91. Springer-Verlag, 1988.

[8] K. Sakamura. Design Policy of the Operating System Based on the BTRON2 Specification. In Ken Sakamura, editor, *TRON Project 1990*, pp. 103–118. Springer-Verlag, 1990.

[9] K. Sakamura. Programmable Interface Design in HFDS. In Ken Sakamura, editor, *TRON Project 1990*, pp. 3–22. Springer-Verlag, 1990.

[10] 坂村 健. *BTRON1* 仕様・ソフトウェア仕様書. TRON 協会, 1989.

[11] 坂村 健. メディアと電脳生活. 情報処理学会情報メディア研究会 *1-1*, pp. 1–6, May 1991.

[12] 木村 憲雄, 坂村 健. BTRON EnableWare 関連仕様とその実現. 第 7回 トロン技術研究会, Vol. 3, No. 1, pp. 1–16, June 1990.

[13] 越塚 登ほか. Abstract Event に基づく Interface Manager. 第 *6*回 トロン技術研究会, Vol. 2, No. 4, pp. 1–22, March 1990.

[14] 植松 理昌, 越塚 登, 高田 広章, 坂村 健. BTRON 多国語パネルマネージャ. 第 *10*回 トロン技術研究会, Vol. 1, No. 4, pp. 1–15, July 1991.

A WSDP 上でのプログラミング例

```
do{  /*  Abstract Event ループ */
  if((aevtype = GetAbstractEvent())<0)
     goto ERROR;

  switch(aevtype&OBJCLASS) {
  case CWINDOW:/* window 操作 */
    switch((aevtype & 0x00ff) << 8) {
    case DEL:/* window の消去 */
      again = LOOPEND;
      break;
       :
    };
     :
  case CMENU:/* メニュー操作 */
    switch((aevtype & 0x00ff) << 8) {
    case SELECT:/* メニュー項目選択 */
      switch((UWORD) ((aev.data&MAIN_ME) >> 8){
      case 10:
        /* テキスト作成 */
        _TEXTBOX(tb1, TB_PARTS|P_DISP, \\
                 200, 130, 400, 150, 20, NULL);
        /* Text Box 登録 */
        SetWO(CPARTS, &tb1, sizeof(TEXTBOX),\\
              TAIL,LOCAL);
        break;
          :
      };
       :
    };
     :
  };
}while(again==LOOPAGAIN);
   :
```