

OS/omicon ウィンドウシステム「出島」について

堀 素史, 小松 徹, 並木 美太郎, 高橋 延匡
東京農工大学 工学部

本報告ではヒューマンインタフェースの研究, 開発を支援する環境である「出島」について述べる.
「出島」はOSを含む, 全てのヒューマンインタフェースに関係する資源を「出島」の対象とすることによって, ヒューマンインタフェースの研究, 開発が容易に行える環境と, アプリケーションプログラムへのヒューマンインタフェースの提供を行なう. この環境の実現のために, 以下の機能が提供される.

- (1) ウィンドウシステムによるデバイスインタフェースの抽象化
- (2) 交換可能なヒューマンインタフェース管理
- (3) オブジェクト指向的アプリケーションプログラムインタフェースの提供

これらの機能によって, ヒューマンインタフェースの実験やユーザフレンドリなコンピュータシステムの開発が容易になる.

"DEJIMA" as a research environment for human interface on OS/omicon

Motofumi Hori, Tohru Komatsu, Mitarou Namiki and Nobumasa Takahashi
Faculty of Technology, Tokyo University of Agriculture and Technology
2-24-16 Nakamachi, Koganei-shi, Tokyo 184, Japan

This paper describes "DEJIMA" as a research environment for human interface on OS/omicon.

"DEJIMA" realizes this environment by employing all resources including those the operating system manages.

This environment performs the following three basic functions.

- (1) Unification of device interface by abstracting it.
- (2) Management of exchangeable human interface module.
- (3) Offer of object-oriented Application Program Interface.

These functions provide extensive choices of design alternatives for human interface.

"DEJIMA" should activate researches on human interface. And it is useful to develop user-friendly applications.

1.はじめに

現在、コンピュータの利用はさまざまな分野に及んでいる。従来は、事務計算や科学技術計算などが主体であった。しかし、ハードウェア技術の飛躍的進歩およびそれに伴うコストの低減によって、小型で個人が道具として多目的な用途に利用できるコンピュータ、すなわちワークステーションやパーソナルコンピュータが登場した。

これら個人用のコンピュータは、設計する段階から多目的な用途に用いられることを前提としている。したがって、コンピュータのハードウェア、それをサポートするオペレーティングシステム（以下、OS）は幅広いユーザ層や多目的な利用に対応する機能を備えることになった。

具体的には、ハードウェアではビットマップディスプレイやマウスなどのポインティングデバイスの装備が特徴である。また、ソフトウェアにおいては、ウィンドウシステム上でメニューやアイコンなどの対話管理を実現しレスポンスの高速化、リアルタイム性を重視したシステム設計が特徴である。これらの特徴によって、エンドユーザがインタラクティブな直接操作を行なうことが可能となる。

このような状況下では、人とコンピュータ間のコミュニケーションを司るヒューマンインタフェースがコンピュータをユーザフレンドリにするかどうかの鍵となる。

しかし、ユーザフレンドリなヒューマンインタフェースを提供するという問題は容易に解決できるものではない。実現までには、認知工学でのヒューマンインタフェースモデルの研究を必要とするが、これらの研究には実験や検証が不可欠である。したがって、実験や検証を容易に行なえる環境が必要となる。

ヒューマンインタフェースは、デバイスのインタフェース、操作方法、アプリケーションプログラム（以下、AP）に提供するインタフェース、APの構築モデルなどさまざまな機能と関係がある。これらの機能はコンピュータシステム全体にわたり、コンピュータシステムのアーキテクチャに深

く影響を与える。

「出島」は、これらヒューマンインタフェースに関係するすべての要因をサポートの対象とし、コンピュータシステム全体でヒューマンインタフェースの研究を行なうための環境である。本報告では、「出島」の概念およびそのデバイス管理機構であるウィンドウシステムについて報告する。

2.ヒューマンインタフェースの研究、開発環境

2.1 User Interface Management System

ヒューマンインタフェースの研究および提供のための環境の必要性が高まってきた結果、これらを扱う環境の開発が始まり、User Interface Management System（以下、UIMS）が1982年に登場した [1]。

UIMSは、APのヒューマンインタフェース部分を開発するためのツールおよびヒューマンインタフェースを提供する機構からなる。UIMSは、AP間におけるヒューマンインタフェースの一貫性、初心者から上級者まであらゆる層のエンドユーザへの対応、エラー処理のサポートを指向して設計されている。

2.2 既存のUIMSの問題点

UIMSによって、APからヒューマンインタフェースの記述を独立させ、ヒューマンインタフェースの専門的な研究および提供を行なう環境が実現された [1]。しかし、既存のUIMSはその実行環境において幾つかの問題点を抱えている。これらの問題は、**コンピュータシステムのヒューマンインタフェースという資源を管理する機構**としてUIMSが位置付けられているのではなく、単なるライブラリとしてユーザに提供されていることに起因する（図1）。以下に具体的に問題点を示す。

(1) UIMS間におけるヒューマンインタフェースの互換性の問題

ヒューマンインタフェースの提供をライブラリの形で実現することは、その構築や提供をOS開

発者以外のユーザに開放することになる。この結果、同一システム上に複数のUIMSが提供される可能性が生じる。

例としてUNIX上で多くのUIMSが提唱されていることが挙げられる[2]。これらのUIMS間にはヒューマンインタフェースの互換性に関する定義はなく、したがってこれらのUIMSを利用して開発されたアプリケーション間にはヒューマンインタフェースの互換性はない。ソフトウェア工学的見地から見れば、多くのシステムが実現され、その使用経験から淘汰されることは望ましい。しかし、UIMSのように資源管理を機能として提供するシステムが複数存在することはエンドユーザに対して複数の操作系を提供することになる。また、AP開発者にとってもどれを採用するかに関して判断に迷うことになる。

(2) 資源管理を行なうOSとの機能的融合の欠如

UIMSがヒューマンインタフェースすべてに対しての一貫性を提供するためには、デバイスがエンドユーザに対して提供するインタフェースまでを対象として管理する必要がある。デバイスの管理は本来、OSが提供する機能である。したがって、UIMSはOSとの機能的融合を行ない、コンピュータシステム全体を対象とした環境となるべきである(図2)。UIMSがライブラリとして実現されている段階では、ヒューマンインタフェースの研究、提供をする環境としては不十分である。

(3) 資源の保護

UIMSは、OSがコンピュータの資源管理を行っているのに対し、AP間で共有するヒューマンインタフェースという資源を管理している。資源管理はOSの主要な仕事である。そのような機能をUIMSが提供するのにAPと同じレベルでしか動作できないとすると、APに対しての保護機能は実現できない。したがって、資源の保護の面から考えると適当ではない。

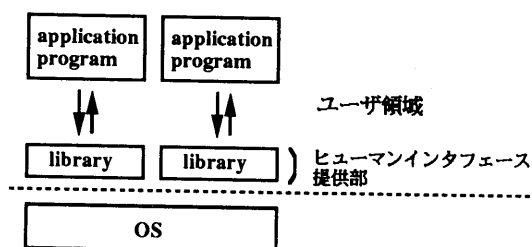


図1. ライブラリとしてヒューマンインタフェースを提供するモデル

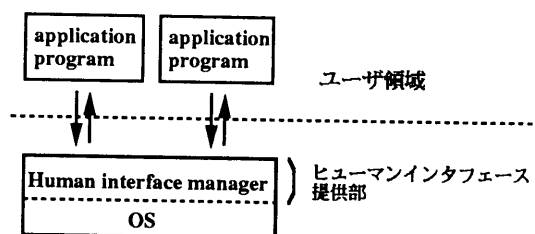


図2. OSの機能としてヒューマンインタフェースを提供するモデル

3. ヒューマンインタフェース研究支援環境「出島」

3.1 「出島」の設計思想

「出島」は、既存のUIMSがもつ上記の問題点を考察し、ヒューマンインタフェースの研究を支援する環境を実現するために以下の設計方針を採用した。

(1) デバイス特性の抽象化によるヒューマンインタフェースの改善

人とコンピュータとのインタフェースの中で物理的な結び付きを持つのはデバイスのインタフェースである。人はデバイスに対して入力の実行を行ったり、出力の内容の確認を行なう。ヒューマンインタフェースに関係するすべての要因の改善を行なうために、デバイスのインタフェースの

改善を行なう。

このインタフェースは、マシン間における互換性が一番重要である。しかし、デバイスは各マシンごとに固有のものを採用している場合がほとんどで物理的互換性は保証されない。例えば、ビットマップディスプレイは多くのコンピュータシステムに採用されている。しかし、解像度や呼び出しのプロトコルに互換性がないためマシン独立なアプリケーションの構築は難しい。

したがってデバイスの特性を抽象化し、マシン独立なインタフェースを提供することによってヒューマンインタフェースの改善を行なう。

(2) ヒューマンインタフェースの研究者が容易に研究を行なえる環境の提供

ヒューマンインタフェースの研究は使用経験からのフィードバックが重要である。したがって、実験のターンアラウンドタイムの短い、フィードバックのかけやすい柔軟な構成とする。

(3) AP開発者に対し、負担が少なく統一のとれたプログラミングスタイルの提供

AP開発者にとっては、ヒューマンインタフェースの部分をその都度構築するのは非効率的であ

る。したがって、共用できるような箇所はなるべく統一してモジュール化し、アプリケーションプログラムインタフェース（以下、API）の形式で提供する。

3.2 「出島」の提供する機能

前節の設計思想に基づいてヒューマンインタフェース支援環境「出島」を実現する(図3)。「出島」では以下に述べる機能を提供する。

3.2.1 抽象化されたデバイス管理によるインタフェースの統一

コンピュータにおける資源の一つであるデバイスの管理を「出島」の概念で統一的行なう。ヒューマンインタフェースの研究や開発を行なうためには、その要素であるデバイス自身の特性も研究の対象とすることが望ましい。デバイスの特性は、操作性を良くする上で一番重要な要因であるからである。また、インタフェースを統一することにより、デバイスのインタフェースの互換性を異なったマシン間で確保することが可能となる。

したがって、「出島」ではデバイスの特性の抽象化による操作性の改善や、マシン間の互換性を確保するためにビットマップディスプレイのフ

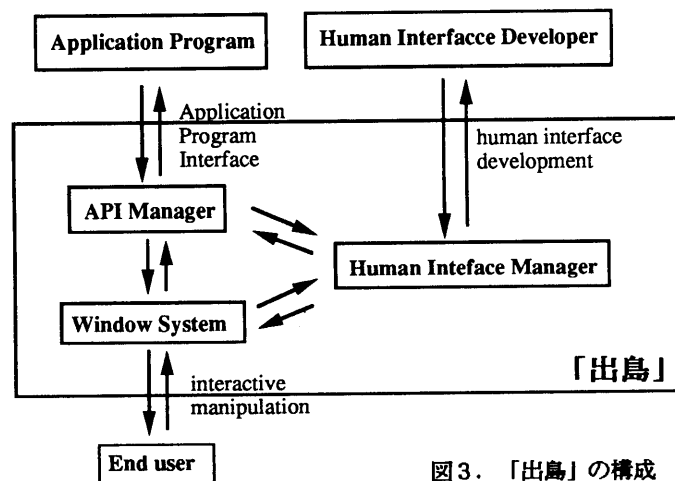


図3. 「出島」の構成

レームメモリやキーボードのコードなどのデバイスのインタフェースを抽象化し、ヒューマンインタフェース構築の際におけるプログラムの負担の減少を図る。

また、インタフェースが類似しているデバイスはさらに高度な抽象化をして同じグループのデバイスとして扱えるようにする。

デバイス管理はウィンドウシステムと呼ばれ、4章でさらに述べる。

3.2.2 交換可能なヒューマンインタフェース

ヒューマンインタフェースを司る部分はモジュール化して交換が可能な設計とする。ヒューマンインタフェースを構築するツールは、このモジュールを入れ換える作業を行なうことによって実現する。この結果、ヒューマンインタフェースの研究者はAPに手を加えることなく、この部分を再構築するだけで新しいヒューマンインタフェースに変更することを可能にする。

ヒューマンインタフェースの提供とは、単に抽象化されたデバイスのインタフェースの提供を意味するのではなく、デバイスのインタフェースとともに手続きを定義し、デバイスに対する操作を記述することである。

例えば、メニューのヒューマンインタフェースモジュールは、メニューに対するキーボードやマウスなどのイベントを受取り、必要な画面表示などの操作を行ない呼び出しプログラムに対してどのような要求が行なわれたかを返す作業までを記述する。

このようにヒューマンインタフェースを提供する部分をヒューマンインタフェース管理部と呼び、後で述べるAPI管理部とウィンドウシステムの間の層で実現される。

3.2.3 オブジェクト指向的APIの提供

オブジェクト指向的APIとは、ヒューマンインタフェースモジュールをオブジェクトとして扱い、APに対して独立な環境を提供するためのプログラム間のインタフェースである。ヒューマン

インタフェースモジュールの定義がオブジェクト指向のクラス定義となり、モジュールを呼び出したプログラムに対してインスタンスを生成し、このインタフェースをAPIとして提供する。この結果、APは独立した入出力環境を得ることが可能となり、他のAPとの関係を意識する必要がなくなる。この機能を提供する部分をAPI管理部と呼ぶ。

メニュー操作のヒューマンインタフェースモジュールを例にする。

AP、つまりAPI管理部を呼び出すプログラムがメニューの初期化を要求すると、API管理部はヒューマンインタフェース管理部のメニューの手続きに基づいてローカル変数を持ったインスタンスを生成し、APに提供する。AP側はこの環境を自分専用のものとして利用し、操作を行なう。

4. 「出島」ウィンドウシステム

4.1 「出島」ウィンドウシステムの設計方針

「出島」ウィンドウシステムは「出島」環境の中でデバイス管理を行なう部分である。このデバイス管理はデバイス独立なインタフェースの提供を設計の基本方針としている。このインタフェースは、エンドユーザに対してのインタフェースとヒューマンインタフェース記述者に対してのプログラムインタフェースの両方の意味を持つ。

4.1.1 デバイスインタフェースの抽象化

「出島」環境では、3.2.1で示したように可能な限りデバイスのインタフェースを統一することによって、ヒューマンインタフェースの改善を行なう。

具体的には、ビットマップディスプレイやLBPなどのビットマップ型のデバイスの扱いを統一し、すべておなじ種類のビットマップ型のデバイスとして扱う方法がある。

既存のOSのデバイス管理においては類似したインタフェースを持つデバイスであっても、後付けのために全く違ったデバイスとして管理されているものが多い。例えば、ビットマップディスプ

レイとLBPのようなデバイスにおいては出力先がディスプレイであるか紙であるかの違いだけでエンドユーザが受け取る情報としては同種のものである。ユーザは、出力する対象が変化するだけに全く異なった操作を要求される。また、ビットマップディスプレイ上に出力されるものと全く同じものを常に得られるとは限らない。

このような問題を解決するために、「出島」ウィンドウシステムはデバイスのより高度な抽象化を行なう。

4.1.2 統合環境におけるデバイス管理

コンピュータをパーソナルな目的に利用する場合、幾つかの作業を並行して行ない、それらの結果を相互に利用する形態が考えられる。例えば、DTPのような目的に利用する場合、テキストエディタ、図形エディタ、グラフィックエディタ、レイアウトツールのAPを起動し、それらを効率良く利用することによって出力を得るような仕事を想定してみる。マルチタスク環境下で複数のAPが動作する場合に重要なことは、デバイスなど限られた資源を効率良く管理することである。したがって、各APに対して、互いを意識しない独立なデバイス環境を与えることがまず必要であり、そのためにウィンドウを利用した入出力の多重化の実現がウィンドウシステムで行なわれている。

4.2 ウィンドウシステムの提供する機能

前節の設計方針に沿った結果、「出島」においてデバイス管理を行なうウィンドウシステムは、以下の機能を提供する。

(1) 構造化オーバラッピングウィンドウ

ウィンドウシステムは、オーバラッピングによるウィンドウ管理を採用した。ウィンドウシステムがよりデバイス独立な環境を提供するためには自由度の高いオーバラッピング方式が適当である。

また、ウィンドウは構造を持たせ、グループ化やグループの親子関係の定義を可能にする（図

4）。

グループ化とは複数のウィンドウをまとめて一つのグループを構成することである。グループ化のメリットの例を挙げる。ウィンドウのスクロールバーやタイトルバーなどの装飾をそれぞれウィンドウとして扱い、これらをまとめて一つのグループとする。この方式ではスクロールバーやタイトルバーをウィンドウの特別な属性として扱う

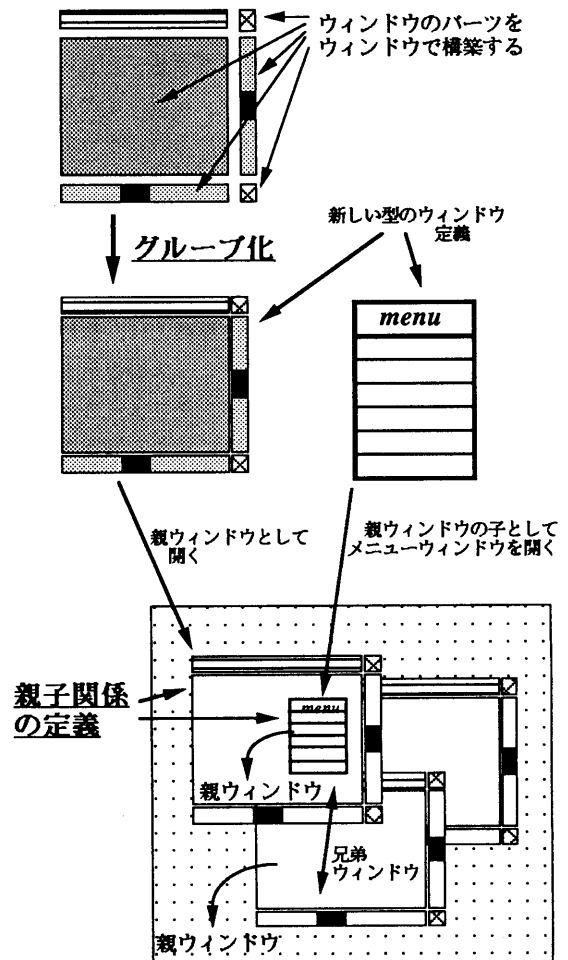


図4. 構造化ウィンドウ

のではなく、独立したウィンドウとして扱う。この結果、ウィンドウにタイトルやスクロールバーのような飾りを定義するための特別な機構を設けることなくさまざまな形態のウィンドウを定義することが可能になる。また、タイトルやスクロールバーも普通のウィンドウと同じ扱いとなるので、ヒューマンインタフェースの研究を行なう際に変更が容易になる。

親子関係は、グループ間で定義でき、生成関係や座標系を引き継ぐ。APの終了に伴う全ウィンドウの消去などは、グループの親ウィンドウを消去するだけで完了する。

APが複数動作するマルチタスク環境下では、AP単位で全ウィンドウを操作できることはウィンドウの管理の点で有効である。

(2) マルチデバイスウィンドウ

現在のウィンドウシステムの多くがビットマップディスプレイが接続されていることを意識しているのに対し、マルチデバイスウィンドウはビットマップディスプレイやそれ以外のビットマップ型のインタフェースを持つデバイスであるタブレットやLBPをウィンドウシステムの枠組の中で同種のデバイスとして扱うものである。

この結果、ビットマップディスプレイ上に表示している内容をLBP上で出力することや、タブレットからの座標入力をフレームメモリからのデータ参照の形で行なえるなど、エンドユーザに特別なデバイスであることを意識させないで利用することが可能となる。

また、これらのデバイスの管理を一つのウィンドウシステム上で実現するので、マルチディスプレイなどもこの枠内で実現が可能である。したがって、ウィンドウとフレームメモリのマッピングの組み合わせによって、今までとは違った統合環境の実現が可能となる。

例として、テキスト、グラフィック、図形のエディタとレイアウトツールによるDTP環境を考

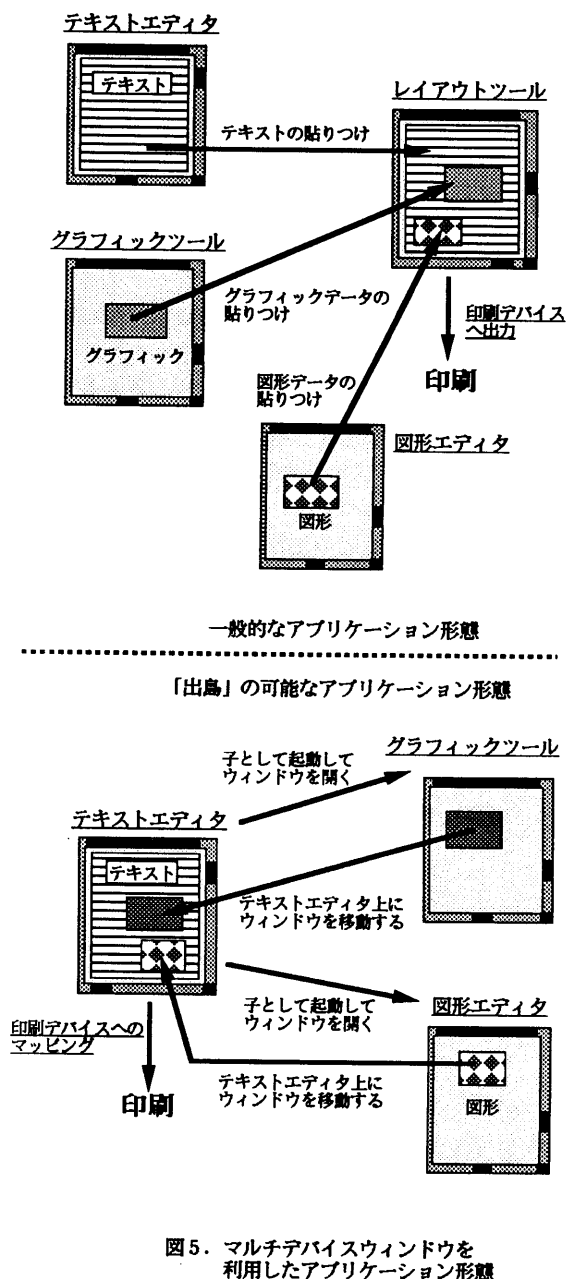


図5. マルチデバイスウィンドウを利用したアプリケーション形態

える (図5)。

一般的には、それぞれのエディタでデータを作成し、レイアウトツールがこれらのデータを読み込み、インタラクティブな操作でレイアウトを行ない、ビットマップディスプレイ上に表示を行な

うことによって出力が得られる。紙に出力を得る場合には、このデータをLBPに転送することによって出力が得られる。この環境においては、レイアウトツールが要のツールとなり、あらゆるツールのデータフォーマットを理解し、展開する機構を持たなければならない。

「出島」ウィンドウシステムでは、どれか一つのエディタ、例えばテキストエディタがレイアウトツールの機能のうち他のデータの表示用領域を確保するレイアウト機能さえ実現していれば、この領域に他のエディタの出力用ウィンドウを挿入すれば表示が得られる。さらに、このテキストエディタのフレームメモリをLBPにマッピングすれば紙に出力が得られる。

この結果、レイアウトツールはテキストエディタに吸収され、データフォーマットを理解し描画する機能の必要はなくなる。

以上のように「出島」ウィンドウシステムは、ユーザとプログラマに対しより柔軟なインタフェースを提供することが可能である。

5. 「出島」およびウィンドウシステムの実現

「出島」は、OS/omicon第2版上で実現した(図6)。OS/omiconはシングルユーザ、マルチタスクのOSでタスクフォースを実行環境として備えている[3]。タスクフォースは、その内部にタスクを複数個生成することが可能である。これらのタスクは、手続きや静的データを共有した実行環境を持つ。この実行環境を利用することによって、オブジェクト指向におけるインスタンス生成機能を記述することが可能となる。「出島」のAPI管理部は、タスクフォースを用いることによって、独立した実行環境をプログラマに提供する機構を実現した。

「出島」ウィンドウシステムは、OS/omicon第2版の機能拡張の形で実現した。この結果、ウィンドウシステムが搭載されたシステムでは、ウィンドウシステムが標準の入出力環境となる。

6. おわりに

コンピュータがより身近な存在になるにしたがい、ヒューマンインタフェースの研究や提供を行なう環境の必要性は高まるであろう。これらの環境を提供する側は、ヒューマンインタフェースに

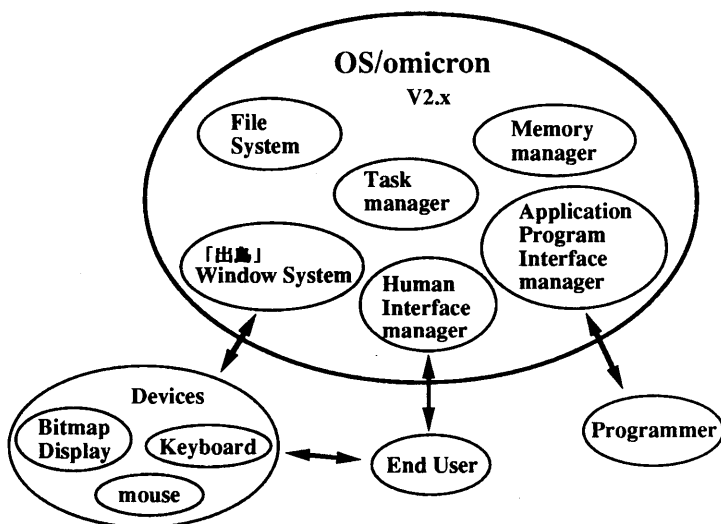


図6. 「出島」の実現

関係するすべての要因を考察し、より広い視野でこの環境を捉えて行かなければならない。したがって、ヒューマンインタフェースの研究や提供を行なう環境の開発には、広い視野にたち、OSをも含めたコンピュータシステム全体を考察することが必要となる。

「出島」はOSのデバイス管理からAPのプログラミングモデルまで広い範囲にわたり、環境としての定義を行なうことによって環境全体としての一貫性を保つことを可能とした。

「出島」の実現により、ヒューマンインタフェースの研究を環境として提供することが可能となった。このような環境の実現は、ヒューマンインタフェースの研究のアクティビティを高くし、ユーザフレンドリなアプリケーションの開発に有利となる。その結果、コンピュータが人間の知的生産のための道具として、より普及することであろう。

参考文献

- [1] Jonas Lowgren, "History, State and Future of User Interface Management Systems", ACM SIGCHI Bulletin, Vol. 20, No. 1, pp. 32-44, 1988. 7.
- [2] H. Rex Hartson and Deborah Hix, "Human-Computer Interface Development: Concepts and systems for its management", ACM Computing Surveys, Vol. 21, No. 1, pp. 5-92, 1989. 3.
- [3] 鈴木茂夫, 他: "OS/omiconにおける日本語プログラミング環境", 情報処理学会論文誌, Vol. 30, No. 1, 1989. 1.