オペレーティング・システム　52－5
（１９９１．９．６）

異機種分散環境におけるUI実現方式

岩崎　未知
日本電気(株)　C&Cシステム研究所

異機種分散環境におけるユーザ・インタフェース・マネージメント・システムのモデルを提案する。このモデルでは、UI実現部分とアプリケーション本体を分離し、異機種上に配置する。両者を結合する方法として、オブジェクトとして表現されたアプリケーション・インタフェースと機種に依存しない形式で記述されたUI定義を用い、これにより機種の相違を吸収する。さらに、操作に対するフィードバックに必要な情報をUI実現部に保持することにより良好なフィードバックが得られ、UIのカスタマイズも容易に実現可能となる。また、このモデルに基づくシステムを試作した。

Creating User Interfaces in Heterogenious Distributed Environment

Michi IWASAKI

C&C Systems Research Laboratories,
NEC Corporation

This paper proposes a model of user interface management systems designed to be used in heterogeneous distributed environment. First, shared application interface objects enable to separate the user interface component from the functionality component of an application. Second, interface specifications described in a set of machine independent forms are used to connect both components realized on different hosts. This approach leads to more modifiable user interfaces. The implementation of a prototype system is also mentioned.

はじめに

コンピュータの応用範囲の拡大と利用形態の多様化により、ユーザのニーズを単一の機種で満足させることが難しくなっており、目的に応じて複数のコンピュータ・システムを使い分けることもしばしば行なわれている。

一方でワークステーションの普及やパーソナル・コンピュータの高機能化によりウィンドウ・システム上での Graphical User Interface(GUI) の利用やネットワークを通じてのコンピュータ利用も一般的なものとなってきた。今後もこの傾向は続き、コンピュータ・ネットワーク上で特定のサービス (アプリケーション: AP) を提供するマシン (AP サーバ) と、ユーザに UI を提供するマシン (UI サーバ) とが協調して問題を解決していく環境が一般的になると思われる。例えば、WS を UI サーバとし、汎用機を AP サーバとしてその上のデータベースの利用を行なうといった利用状況が考えられる。

これに対して、エンドユーザやプログラマがシステムの違いに悩まされる局面も増加している。例えば、OS 毎に異なる資源アクセス・インタフェースや、アプリケーション毎に異なる UI に対応しなければならない。

この問題を解決するためには、異なる種類のコンピュータ・システムが相互に有機的に接続され、互いに「システムの違いを意識する必要無しに他のシステム上のサービスを享受」でき、また「UI を自由に選択 / 変更可能」な環境の構築が必要である。この要求に応えるため、現在、OS 側 [6] 及び、UI 側 [2] から検討を進めている。OS 側では主に異機種間での資源の相互アクセスを支援する。一方、UI 側では AP 本体からの UI 実現部の分離と、両者の異機種上での動作、そして UI の自由な変更を支援する。

本稿では、UI からのアプローチから異機種分散環境における User Interface Management System(UIMS) のモデルを提案し、その上での UI の実現方式と試作システムに関して述べる。

1 設計目標

前記の環境でのシステムに対する要求を満たすため、以下の機能の実現を目標とする。

•UI 部と AP 本体の物理的な分離 UIMS は本来 UI と AP 本体との論理的な分離とを狙っているが、ここでは異なるホスト上に物理的にも分離する。また、エンドユーザにも利用可能な様、通信、同期といった概念の詳細をシステムに隠蔽する。

•UI 部と AP 本体の異機種上での動作 機種の相違を吸収し、異機種間での情報伝達を可能とする。特定のシステムに固有の概念は隠蔽する。

•AP 記述言語に依存しない 機種毎に利用可能な言語が異なるため、なるべく利用言語に依存しない枠組とする。既存 AP への適用も考慮する。

•UI 部と AP 本体の実行時結合 AP 実行中の UI 変更と実行の継続を可能とする。また、UI 部と AP 本体の自由な組合せを許す。

•UI サーバでのフィードバック実現 ユーザの操作に対するフィードバックを UI サーバで行ない、AP サーバに GUI の高い応答性を実現するための負荷をかけず、高速なレスポンスを実現する。

•UI エディタの提供 UI の変更は画面上で実行可能とする。簡単な変更では、テキストによるコーディングを不要とする。個々の AP から独立した UI 変更の枠組を提供する。

2 モデル概要

2.1 オブジェクトによる構成

アプリケーション・プログラムは、一般に、[図 1] の様な構造として捉えることが出来る。 ここ

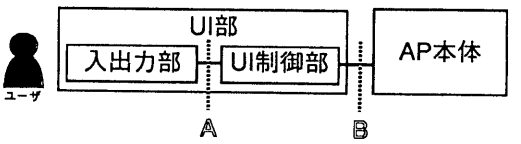

図 1: アプリケーションの構造

で、フィードバックをなるべく UI サーバのみで行なうために、図中、左から B の部分までを UI 部と

して分離しUIサーバ上で実現することを提案する。

左からAの部分までをAP本体から分離しているものとしてX-Windowがあるが、それ自体では今回の要求にはそぐわない。X-Windowに用意されているプリミティブは抽象度が低く、UIに関してAP本体としてのクライアントが責任を持つ形になっているためである。

Bの部分で分離するために、ここでは、UI部とAP本体が共有のデータを通して情報を交換するモデル[1]を分散環境に適合する様に拡張する。物理的に分離するために、共有データを分散共有オブジェクトの集合とし、これをアプリケーション・インタフェース(API)とみなしてAPIオブジェクトと呼ぶ。

APIオブジェクト群によってUI部とAP本体を分離することにより、[図2]の様に、AP全体を、UI部、2つのAPI部、AP本体の4つのコンポーネントで構成する。またUIエディタを用意する。

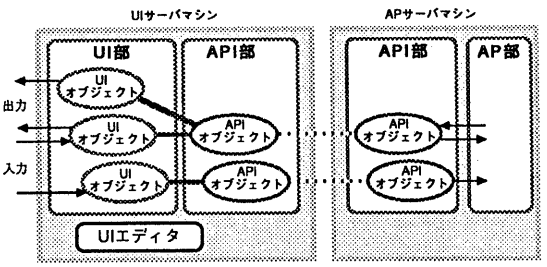

図2: APIオブジェクト群による分離

分散共有オブジェクトとしてのAPIオブジェクトのペアの間では自動的に一貫性を維持する。UI部は、UIオブジェクト群をAPIオブジェクトに結合することで実現する。APIオブジェクトは、AP内の、外部からのアクセスを許す部分をオブジェクトとして外部にexportしたもの(後述)であり、AP本体が内部データ等をオブジェクトとしてexport出来る必要がある。AP本体は、全体の中では「AP固有の機能を実現するオブジェクト」といった位置付けとしている。

2.2 動作

UIオブジェクトはユーザからの入力をイベントとして受けつけ、解釈した後、他のUIオブジェクトやAPIオブジェクトにメッセージを送る。APIオブジェクトはAP本体に通知する。一方、AP本体内部の状態変化は必要に応じてAPIオブジェクトに通知され、それをUIオブジェクトがユーザに表示する。UI部とAP本体は非同期に動作する。

2.3 UIエディタ

UIエディタは、UIオブジェクトの生成/削除、オブジェクト間の関係変更機能を提供する。これにより、例えばユーザのレベルや嗜好に合わせて表示する項目やレイアウト、あるいは入出力形態を選択することが可能である。APIオブジェクト自体の生成/削除は行なわない。よって、UI編集時にもAPIオブジェクトは動作可能であり、APIオブジェクト間の一貫性維持を同期型通信で行なってもAPサーバが待たされることはない。

オブジェクト間の関係の変更は、2つのオブジェクトの間に方向をもったリンク(制約)を付加/削除することで行なう。リンク生成時には、結合する2つのオブジェクトのタイプにより適切なメッセージの形式を予め用意されたデータベースから検索する。結合に意味の無い場合はエラーとする。UI編集時には、UIエディタがAPIオブジェクトやオブジェクト間の関係を示すリンクを画面上に表示し、その画面上での操作を可能とする。変更後、リンク情報も含めてオブジェクトの属性を保存する機能を持つ。

3 UI部とAP本体の分離

3.1 APIオブジェクト

提案方式では、UI部とAP本体の内部での変化はAPIオブジェクトを経由してのみ伝達される。これによりUI部の変更の影響はAP本体に及ばない。また、UIオブジェクトもAP本体も自ホスト

に存在するAPIオブジェクトの操作のみを行なえば良い。これによりユーザ(エンドユーザとアプリケーション・プログラマ)から(異機種間)通信を隠蔽している。

APに対して何をAPIオブジェクトとして選択するかは、例えば以下のものが考えられる。

- ドローイング・ツール: 画面上に表示されたオブジェクト群のそれぞれ
- 画面上のフィールドを埋めていき、実行キーを押すタイプのUIを備えたAP: 各フィールドの値と実行キー

APIオブジェクトとしては、任意の構造を持ったオブジェクトが利用できるべきである。これは、「単純なオブジェクト + それを構造化する枠組」で実現する。複合(構造化)オブジェクトの構築/利用手法に関してはここでは扱わない。単純なオブジェクトである“value”と“trigger”に関してのみ議論する。

単純なものとして、比較的単純な型の「値」を一つ保持するオブジェクトを考える。これは、「変数」に操作用メソッドを加えたものに相当する。値の伝達を目的とするものなので、「値のset/getを行なうメッセージを受け付け、値がset(変更)された時に他のオブジェクトにその値を通知すること」を基本機能とする。そこで、これを基本的なtypeである“value”として用意し、この組合せで全てを表現することを基本方針とする。“value”はset可能な値の区間または集合を持ち、setすべく指定された値が受け入れられないものであれば、値は変更せず要求元に通知される。

AP本体が通常の手続き型言語で記述されている場合、“value”は変数に対応させることが出来る。手続きの起動もこれで可能であるが、便宜を図るため、タイミングを通知するオブジェクトとして“trigger”を設け、手続き起動に用いる。

なお、APIオブジェクトは膨大な数になり得るので、ユーザのブラウジングを支援するため、tree形式に構造化(階層化)する機能を用意する。

3.2 UIオブジェクト

UIオブジェクトはX-Toolkitのwidgetに近いイメージのものであるが、決まったパターンの動作を自律的に行なう能力を備えており、画面上に配置すればAP本体の助け無しに動作可能なものとする。

APIオブジェクト同様tree形式に構造化可能とする。UIオブジェクトでは、これをUI編集時の操作性向上のため画面上でのオブジェクトのグループ化機能として利用する。また、表示の際の階層関係も構造として保持する。

3.3 AP本体

AP本体がオブジェクト(群)としての形態をとって実現されていれば、それほどの問題なくAPIオブジェクトと結合可能である。通常の手続き型言語で記述されている場合には、一つのオブジェクトとして見せるようにする。この場合、AP本体がexportするデータに対する内部データを操作するメソッドとしてのset/getメソッドと、手続きを起動するメソッドを用意する。その変化をAPIオブジェクトに通知する必要があるAP本体内データについては、やはりsetメソッドを用意し、その内部からAPIオブジェクトに通知する。これらでの引数の渡し方も決める必要がある。イベント駆動パラダイムで記述されていない場合、イベント(待ち)・ループを外部に用意し、この部分でAPIオブジェクトからメッセージを受けとり、その内容にしたがって用意されたメソッドを呼び出す。

4 UI部とAP本体の実行時結合

4.1 起動

APの起動は、UI部を実現するUI実現プログラムから行なわれる。UI実現プログラムはエンドユーザに対するコマンド・インタプリタとして予め起動されているものとする。ユーザがUI実現プログラムに対して何らかのAPの起動を指示すると、UI実現プログラムはUNIXのremote shell

に近い手順でAPサーバ上のAP本体を起動する。

4.2 インタフェース定義の転送と部品の利用

UI部を独立させるため、UIオブジェクトの実体はUIサーバ上で生成する。APIオブジェクトの実体は両サーバで生成する。このために、各サーバ上に各種オブジェクトの実体を生成するための「標準化された部品セット」を用意する。

起動時の全体の構成は、[図3]の様なものとなる。アプリケーション毎に予め記述されたAPI定

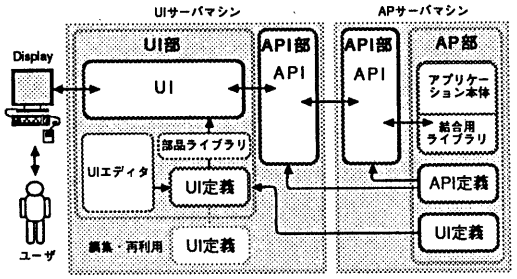

図3: 全体構成

義とデフォールトのUI定義をAPサーバからUIサーバへと転送し、これらインタフェース定義からオブジェクトの実体を生成する(定義の記述形式は後述)。APIオブジェクトは両サーバで同一の定義から生成されるため必ず対応をとることが出来る。

オブジェクトは、インタフェース定義を解釈可能なものとして機能仕様は標準化されるが、各部品の実際の実現手段は各サーバに任される。

インタフェース定義を1度転送すれば、次回、同一のAPを起動する際はそれを利用できるため転送は不要となる。

UIの変更は、UI定義の編集/保存として実現される。ユーザ毎にこれを保存するため、ユーザとAPにそれぞれユニークなIDを付与し、変更後のUI定義にこれら2つのIDをつけて保存する。

4.3 インタフェース定義

オブジェクトの属性には次のようなものが存在する。ID、タイプ、名前、位置、大きさ、グループ化等のtree構造を保持するためのポインタ、動作時に自分が依存関係をもつオブジェクト群へのポインタ、操作可能か否かを示すフラグ等である。

UI定義とAPI定義はこれらを記述したものとして、例えば以下の形式で保存される。

```
((ID publicID objectType objectName
  (model view controller))
 ((groupParent groupSibling groupChild)
  (viewParent viewSibling viewChild)
  (dependencyParent dependencySibling
    dependencyChild))
 (positionX positionY width height
  ( ... ))
 ((target1 selecter1 parameter1)
  (target2 selecter2 parameter2) ... )
 ( ... ))
```

これがUIあるいはAPIオブジェクトの一つに対応する。ここでは、tree構造の表現には3つのポインタを使用し、それぞれ、親、妹、長女を指している。各オブジェクトは、自分に変化が起きた時、targetで指定されたオブジェクトに対してselecterで指定されるメッセージを送る。

4.4 分散共有オブジェクト

分散共有APIオブジェクトのペアの間では一貫性が維持され、一方を操作すれば他方に伝達される。このペアはmasterとslaveという形で実現した。あるオブジェクトを3つ以上のホストで利用する場合にはslaveを増やしていく。APIオブジェクトに対する操作はまずmasterに通知され、その後、各slaveに伝達される。masterはモニタとしての機能も兼ねる。各ホストに実体が存在することにより、UIオブジェクトやAP本体を結合する実体が常にローカルに存在することになり、UIの編集を始めとする処理を容易なものと出来る。

対応するAPIオブジェクト間の参照は、APIオブジェクトのmasterとslaveに同一かつ不変のIDを付与することで実現する。排他制御等を厳密に行なう必要のあるAPでは、オブジェクトにlocked

属性を設け、ロック制御プロトコルで制御する。

オブジェクト間の通信は、値をsetするメッセージによって行なわれる。最低限、次の情報を伝達できれば良い。

```
(set publicID value)
```

5 異機種間通信

異機種間通信機能としては、次の処理を実行可能なものを用意する。AP本体の起動、通信路確保、インタフェース定義のダウンロード、共有APIオブジェクトの一貫性維持である。これを実現するためには、以下のものが必要となる。

- 共通のデータ表現
- インタフェース定義転送プロトコル
- 共有オブジェクト間一貫性維持プロトコル

なお、資源へのアクセス・メソッドのシステム間の相違に関しては別途解決されているものとする。

6 UI部でのフィードバック

6.1 一般的な問題点

フィードバックを行なうためには、一般に、AP本体に存在する知識が必要であり、これ無しには正しいフィードバックが出来ない。例えば、入力の妥当性を検証して結果に応じたフィードバックを行なうためには、通常、AP本体内部の妥当性検出ルーチンを起動する必要がある。以下ではこれをフィードバック情報と呼ぶ。

一方、GUIにおいてはダイレクト・マニピュレーション型のUIが採用されることが多いが、ダイレクト・マニピュレーションには、継続的なフィードバックが必要である。例えば、「マウスのドラッギングにより画面上のアイコンを移動させる操作に対し、ドラッギングの途上で現在の移動先が適正か否かに応じて常になんらかのフィードバック(例えばアイコンの反転表示)を行なう」といったUIを実現しようとすると、マウスの移動が起こる度にフィードバック情報の利用が必要となる。

実際に毎回AP本体に通知することに意味がある場合もあるが、そうでない場合には両者の間の交信が増加し、UI部とAP本体が分散して実行される環境では通信の頻度やAPサーバ上でのコンテキスト・スイッチの増大を招いてしまう。また、通信遅延等により高速なレスポンスを実現することが困難となる。さらに、フィードバックの細部をAP本体が決定する枠組では、UIMSの狙いの一つであるUIとAP本体との分離度の向上が本質的に困難である。

6.2 フィードバック情報の伝達

本システムでは、UIサーバのみでフィードバックを行なうため、フィードバック情報をAP本体のみでなくUIサーバ上のUIオブジェクトやAPIオブジェクトにも保持する。

フィードバック情報には、AP実行中に「変化する情報」と「変化しない情報」がある[4] [5]。例えば、コンピュータと人間がチェスを行なう場合、(正しい入力としての)駒を動かせる位置はゲームの進行と共に変化するが、駒を動かす時の表示(アニメーション等)をAP本体がゲーム中に変更することは通常無い。ユーザ側がUIの変更として行なうことはあり得るが、それはUI側で対処すべき問題である。

変化しない情報に関しては、UIオブジェクトによってフィードバックを行なうために、代表的なフィードバックのパターンを極力UIオブジェクト用部品として用意し、適当なパターンを備えたUIオブジェクトを選択することを「変化しない部分」の保持手段として利用する。

変化する情報に関しては、APIオブジェクトで保持するようにし、変化した時点で更新する。チェスの例では、状況が変化した際に、AP本体が駒の移動が可能な位置を再計算してAPIオブジェクトに通知するか、または、APIオブジェクトにこれを判定する機能を持たせることになる。前者の場合、APIオブジェクトの「set可能な値の集合」が更新される。後者の場合、APIオブジェクトで実行可能な判断プログラムを送り込むことになる。

なお、これによって多量のデータ転送が発生する場合の最適化手法として、一貫性維持機構の動作を遅らせることが考えられる。これは、APIオブジェクトのslave群に自分の内容が最新のものか否かを示すflag(valid bit)を用意しAPIオブジェクトのmasterで状態が変更された場合には、slave群のvalid bitのみ変更することで行なう。slave群は必要に応じて、即時、あるいはon demandにmasterの内容を獲得する。内容獲得の時期を決定する適切な戦略はAPIオブジェクト毎に異なり、また、AP実行中にも変化し得る。そこで、APIオブジェクトの属性としてこれを表現する属性を設け値を設定できるようにする。

7 試作システム

動作確認と、用意すべき部品の種類やプロトコルの検討のため、システムの試作を行なった。

7.1 構成

試作システムは[図4]の様な階層として構成されている。UNIXマシン (EWS-4800)と汎用機(ACOS630)の間での利用を想定して設計されており、現在はUNIX間で動作している。入出力には

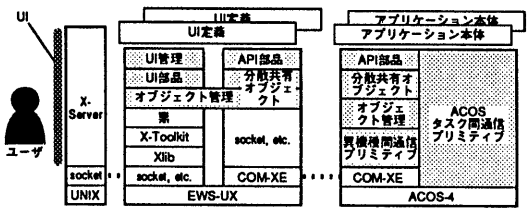

図 4: 構成

X-ToolkitとX上のUI構築ツールである鼎(かなえ)[3]を使用している。

7.2 UIオブジェクトとAPIオブジェクト

オブジェクトは、C言語上で実現した。UIオブジェクトとしては、ボタン、テキスト・フィールド、ポップアップ・メニュー、スライダ(ボリューム)等が用意されている。インタフェース定義は簡略化したものを使用した。

UI編集時には、APIオブジェクト等、通常表示されないオブジェクトも画面上に表示する必要がある。そこで、試作システムでは全てのオブジェクトが表示形態を持つ様にした。システムのモードがUI編集モードになると、通常不可視なオブジェクトは自らの表示機能実現部を活性化する。また、UIオブジェクトは編集操作を受け付けられるよう自分の表示機能実現部を変更する。

試作システムでの「変化する」フィードバック情報の利用は、UIオブジェクトに定められたフィードバック・パターンを生成する上での一部のパラメータを変更できるというレベルに留まっている。例えば、数値入力フィールドやスライダに対して、入力が妥当であるか否かに対応するフィードバックを定めておき、一方、入力として許容する数値の範囲をフィードバック情報としてAPIオブジェクトから供給する。

7.3 通信

通信はsocketとTCP/IPを用いバーチャル・サーキットを張って行なった。データ表現としては文字列を使用し、さらに、そのデータ本来のtypeを表す情報を付加している。

AP本体とAPIオブジェクトを結合するために、AP本体内データ操作メソッド、手続き起動メソッド等を用意し、イベント・ループを含むライブラリとこれらメソッドをC言語上から利用可能な形で作成した。

7.4 利用例

試作システム上で簡単なAP(作業時間管理プログラム)を作成/動作させた画面の例を[図5]に示す。この画面はUIエディタを起動した状態のものである。左下のウィンドウがユーザが通常使うUIであり、内部にUIオブジェクトが見えている。右下はAPIオブジェクトを表示させたものである。これは、UIオブジェクトとAPIオブジェクトの参照関係を変更する時に利用する。画面上で選択

されているUIオブジェクトに対して、対応するAPIオブジェクトとの関係(リンク)が表示されている。右上にAP本体の内容を表示するウィンドウを開いているが、これは内部動作確認のためであり、通常は表示しない。

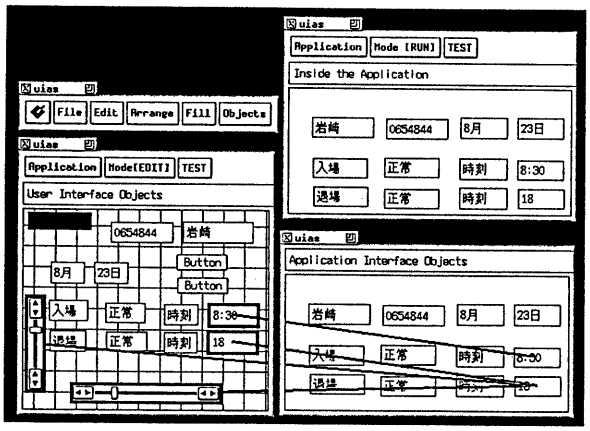

図 5: 利用画面例 (UI編集中)

UIの変更としては、部品の生成/削除、種類、位置、サイズの変更、グループ化、リンクの生成/削除が画面上で可能となっている。リンクの変更は2つのオブジェクト間にマウスで線を引くことで実行できる。また、作成済みのUI定義をいくつも使い分けることや、配布することも可能である。

おわりに

異機種分散環境で動作しUIの容易な変更を可能とするUIMSの枠組と、その上でフィードバックを実現する手法に関して提案した。またこれに基づく試作システムについて説明した。

このシステムを実用にするためには、まだ多くの問題を解決する必要がある。同期、排他、一貫性維持戦略、フィードバック情報の記述形式、UI編集の正当性維持支援、インタフェース定義のバージョン管理、AP本体結合用ライブラリの生成支援、等である。今後は、これら手法の実現形態の検討/改良と共に、構造を持ったオブジェクトの簡便な作成/利用手法、部品自体のカスタマイズによるUIカスタマイズ手法、UI部とAP部との結合の視覚化手法、フィードバック情報の保持手法を中心に検討を継続する予定である。

最後に、本研究の機会を与えて下さると共に、有益な御指導を下さいました日本電気(株)石黒辰雄支配人、C&Cシステム研究所の山本昌弘所長、久保秀士主管研究員、小池誠彦部長、横田実課長、中崎良成課長を始めとし、貴重な御助言を下さいました方々に深く感謝致します。

参考文献

[1] Hudson, Scott E., "UIMS Support for Direct Manipulation Interfaces", ACM Computer Graphics, vol. 21, no. 2, pp. 120-124, (1987)

[2] Iwasaki, Michi(岩崎 未知), "異機種分散環境におけるUIMSの構成", 情報処理学会第41回全国大会講演論文集(5),5-221, 1990

[3] Rekimoto, Jun'ichi(暦本 純一), 垂水 浩幸, 菅井 勝, 山崎 剛, 猪狩 錦光, 森 岳志, 杉山 高弘, 内山 厚子, 秋口 忠三, "エディタを部品としたユーザインタフェース構築基盤: 鼎", 情報処理, vol. 31, No. 5, pp. 602-611, (May. 1990)

[4] Szekely, Pedro, "Modular Implementation of Presentations", CHI+GI'87, pp. 235-240, (1987)

[5] Szekely, Pedro, "Separating the User Interface from the Functionality of Application Programs", no. CMU-CS-88-101, Technical Report, Computer Science Dept., Carnegie Mellon University, (1988)

[6] Takano, Yousuke(高野 陽介), 久保 秀士, "異種OS間連携を支援する拡張名前管理機能", 情報処理学会研究会報告 90-OS-48, vol. 90, No. 67, pp. 77-84, (Sep. 1990)