対話の階層モデルに基づくユーザインタフェース管理システム†

横山孝典††* 谷 正之††
荒井俊史†† 谷藤真也††

応用プログラムからユーザとの対話処理を行う部分を分離することにより高度なユーザインタフェースを効率よく構築できるユーザインタフェース管理システム (UIMS) が注目されている．本論文ではユーザと計算機との対話における情報の変換過程を意味レベル，構文レベル，字句レベルの処理から成る階層モデルにより表現し，これに基づく階層構成を有する UIMS を提案する．本 UIMS は実行時に対話の管理を行う対話処理系とユーザインタフェース構築環境から成る．対話処理系はモジュール性の優れたオブジェクト指向に基づく構成とし，階層モデルにおける意味レベルの処理を実行する機能オブジェクトと構文レベルの処理を実行する表現オブジェクトを基本構成要素とする．なお字句レベルの処理は表現オブジェクトが呼び出すグラフィックサブルーチンが実行する．そしてこれらのオブジェクトの組み合わせでさまざまな形態のユーザインタフェースを実現可能とするとともに，表現オブジェクトの交換によりユーザインタフェースを簡単に変更できる．また表現オブジェクトを木構造に組み合わせることにより，統一的な画面管理やイベント管理を実現した．一方，構築環境は画面上で部品を組み立てる要領で対話的にユーザインタフェースを開発できる UI エディタを有し，構築効率を大幅に向上できる．

1. まえがき

最近計算機の使いやすさが重要視され，メニューやアイコンを使用したわかりやすく対話性の優れたユーザインタフェースが要求されるようになった．このため応用プログラム全体に占めるユーザインタフェース部分の割合が増大し（対話型プログラムの約半分以上がユーザインタフェースに関する処理と言われる），開発コストの上昇を招いている．

この問題の解決のため，1980年代初頭にユーザインタフェース部分を応用プログラム本体から分離し，宣言的言語等による効率的な方法でユーザインタフェースを構築できるユーザインタフェース管理システム (User Interface Management System, UIMS) が現れ，活発に研究が行われるようになった[1)-7)]．

UIMS は，応用プログラム中に直接プログラムコードとして対話処理を記述する従来方式に比較して，①ユーザインタフェース構築の省力化が図れる，②ユーザインタフェースを意識せずに応用プログラムを作成できる，③汎用的なユーザインタフェースを提供できる，④一つの応用プログラムに複数の異なったユーザインタフェースを提供できる，⑤試行錯誤的にユーザインタフェースを構築できる，⑥ユーザインタフェースの修正，変更が容易である，⑦入出力ハードウェアはもちろんグラフィック基本ソフトにも依存しない応用プログラムが開発できる等の利点がある．

以上のように UIMS はきわめて強力な手法として注目され，研究されているが，現在のところその概念はまだ固まっておらず，標準的な構成法も存在しない．

本研究の目的は，ユーザと応用プログラムとの対話における UIMS の位置付けと必要な機能を明らかにし，汎用的で実用的な UIMS の構成法を確立することにある．

UIMS は応用プログラム本体とユーザの間で交されるデータや制御情報の変換処理ソフトウェアと見なすことができる．そこで本論文では応用プログラムとユーザの対話における情報の変換過程を，その情報のレベルに従って階層化する，いわゆる階層モデルにより解析し，これに基づく階層構成を有する UIMS を提案する．これにより応用プログラムからの独立性が高く，ユーザインタフェースの構築や変更が容易であるとともに，さまざまな対話形式やシステム形態に対応可能な UIMS を実現できる．

以下では，まずユーザと応用プログラムの対話の階層モデルについて述べ，次に，UIMS の構成や機能に関する課題を明らかにする．そしてそれらの課題を

† A User Interface Management System Based on the Layered Model of Man-Computer Interactions by TAKANORI YOKOYAMA, MASAYUKI TANI, TOSHIFUMI ARAI and SHIN-YA TANIFUJI (Hitachi Research Laboratory, Hitachi Ltd.).

†† (株)日立製作所日立研究所
* 現在 (財)新世代コンピュータ技術開発機構研究所第5研究室

満足するため，階層モデルに基づく UIMS の構成とオブジェクト指向による実現方式を提案し，その動作を説明する．最後に本 UIMS をエンジニアリングワークステーション上に試作しての評価と今後の展開について述べる．

2. ユーザと応用プログラムの対話

2.1 対話モデル

一般にユーザが計算機と対話を行って仕事をする場合，頭のなかで考えている概念的な仕事（タスク）をその実現に必要な計算機に対する操作に展開していると言われる[8]．また，これとは逆に計算機からの出力がユーザの目的に対してどのような意味をもつかという逆方向の変換も行っている．

一方計算機側でも，ユーザからの入力を解析しそれを応用プログラムが処理可能なデータに変換したり，逆に応用プログラムが扱っているデータをユーザが理解可能な形式に変換して出力したりする．

最近のアイコンやメニュー等を使用した直接操作(direct manipulation)[9]によるインタフェースはユーザ側の変換処理の負担が小さくわかりやすいが，応用プログラム側での変換処理は複雑となる．

このようなユーザと応用プログラムにおける対話情報の変換過程を，処理のレベルに従って階層分割して表現する方法にいわゆる階層モデル[10),11)]がある．階層モデルは対話のレベルに応じたモジュール化手法を示唆することからユーザインタフェースの設計に役立つ．

対話情報変換仮定の階層化の方法にはいくつかの案があるが，対話コマンドの解析については以前より，言語解析と同じように，字句解析，構文解析，意味解析の各処理に階層化する，いわゆる言語モデルがしばしば使用され，ユーザインタフェースの設計に利用されている[4),11),12)]．ここではこの手法をグラフィカルなユーザインタフェースに拡張する．例えばマウスによりメニューを選択し，応用プログラム内の特定の手続きを起動する例について考えると，ユーザのマウス操作からクリックされたボタンの番号や入力座標（論理座標）を算出するのが字句レベルの処理，入力座標から選択されたメニュー項目を求めるのが構文レベルの処理，選択項目に従って応用プログラム内の手続きを起動するのが意味レベルの処理ととらえる．

そしてこの言語モデルにおける階層化手法を前述の階層モデルに適用する．このモデルを図 1 に示す．こ

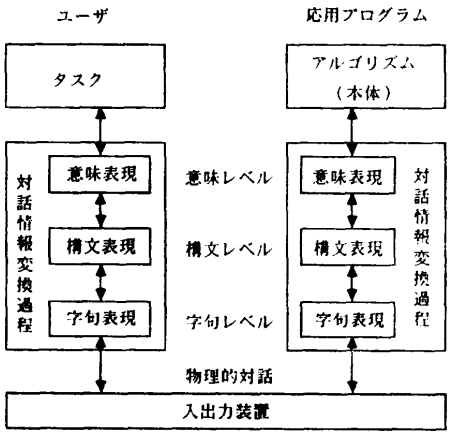


図 1 対話の階層モデル
Fig. 1 Layered model of man-computer interactions.

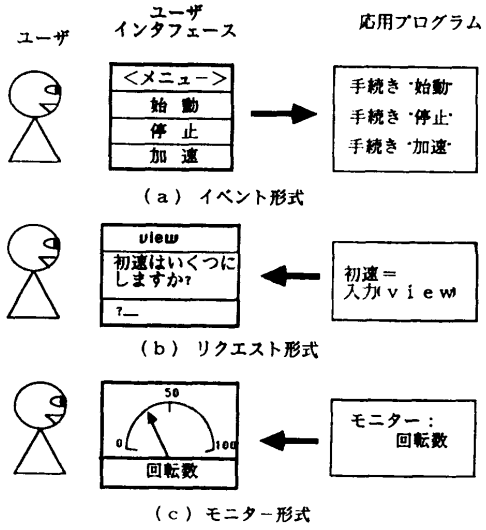


図 2 対話形式
Fig. 2 Styles of interactions.

のモデルではユーザおよび応用プログラムの対話情報を字句レベル，構文レベル，意味レベルに階層化して表現する．ユーザと応用プログラムの対話は，両者の対話情報変換過程と入出力装置を用いた物理的対話を経た，ユーザのタスクと応用プログラム本体間の対話情報のやりとりと見なす．

2.2 対話形式

ユーザと応用プログラムとの対話形式はその制御形態という観点から以下の 3 種類に分類できる．第 1 の形式はユーザが応用プログラムに対して特定の処理の起動やデータ値の変更を要求するもので，これをイベント形式(ユーザ主導形式)と呼ぶ．例えば図 2(a)のようにメニューにより応用プログラム内の所定の手続

きを呼び出す場合である．スプレッドシート上でのデータの書き替えもこの形式である．イベント形式の特徴はユーザ主導型のインタフェースを提供できることである．

第2の形式は第1の形式とは逆に，図2(b)のように応用プログラムが処理の実行に必要なデータの入力をユーザに要求したり，計算結果を出力する場合である．これをリクエスト形式(応用プログラム主導形式)と呼ぶ．一般のプログラミング言語に組み込みの入出力方法 (READ 文や WRITE 文等) はこの形式である．

上記二つの形式のほか，応用プログラムの処理の経過や内部データの値をユーザが一方的に眺めるという，第3の形式がある．これをモニタ形式と呼ぶ．例えば図2(c)のように応用プログラム中の変数の値をメータ表示して観察する場合である．この形式はリクエスト形式と一見似ているが，応用プログラムの処理に影響を与えない点で消極的な対話であり，別の形式として分類する．

実際のユーザと応用プログラムの対話では上記3種の対話形式が混在して使用される．

3. UIMS の課題

3.1 UIMS の基本形態

従来は図 3(a)のように入出力処理を応用プログラム内に直接記述していたが，ユーザインタフェース部分が応用プログラム本体と混然一体となっているためプログラムの訂正や変更が困難という欠点があった．

これに対し UIMS を用いる方法は，図3(b)のように応用プログラムは論理的な処理（アルゴリズム）のみを行うものとし，ユーザとの具体的な対話処理部分を分離，独立させる方式である．

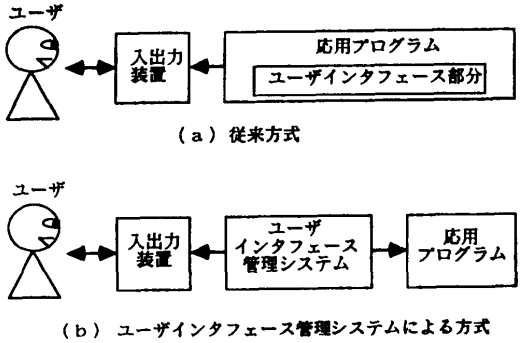


図 3 ソフトウェア構成
Fig. 3 Software structures of application programs.

一般に UIMS は単に応用プログラムの実行時にユーザとの対話処理の管理を行うだけでなく，ユーザインタフェースを構築するためのツールも含む．ここでは前者を対話処理系，後者を構築環境と呼ぶ．

3.2 対話管理上の課題

ここでは UIMS の対話処理系の機能に関する課題を対話形式，データの表現形式，画面管理の各観点からとりあげる．

前述のようにユーザと応用プログラムとの対話形式はイベント形式，リクエスト形式，モニタ形式の3種に大別できるが，実際の応用プログラムではこれらが混在して使用される．したがって対話処理系はこれらすべての対話形式の実現に適した構成とする必要がある．

最近のワークステーションでは複雑なグラフィック処理を駆使して入出力情報をさまざまな形態で表現する．したがってデータの多彩な表現が可能であることはもちろん，他の表現に簡単に変更できなければならない．

また，グラフィックを多用したユーザインタフェースでは多数の表示物が画面上に並ぶことになり，これらの表示物を統一して扱う機能が不可欠である．例えばある表示物を移動した場合それに接続されている他の関連する表示物も移動してやる必要がある．

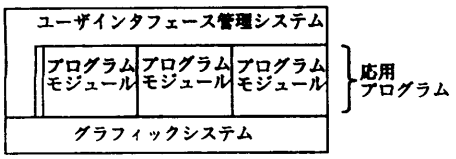
3.3 対話処理系の構成上の課題

対話処理系には，①応用プログラムからの独立性が高いこと，②ユーザインタフェースの部品化がしやすいこと，③変更，修正が容易なこと等が要求され，これらを満足する構成とする必要がある．

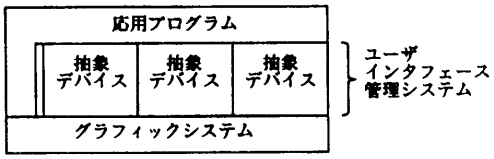
ところで，前述のように UIMS の標準的な構成法はないが，図 4(a)に示すような外部制御型 UIMS と，同図(b)に示すような内部制御型 UIMS が存在することが知られている[2]．

外部制御型では，処理の制御権を有する UIMS がユーザからの要求に応じて複数のモジュールから成る応用プログラムを起動する．したがってイベント形式の対話を支援するのに適している．これに対し内部制御型では応用プログラムが処理の制御権を有し，入出力処理手続きの集合である UIMS を呼び出してユーザとの対話を行う．この入出力手続きは仮想的な入出力デバイスと見なせ，抽象デバイスと呼ばれる．内部制御型 UIMS はリクエスト形式の対話を支援するのに適している．

すべての対話形式を支援するには，両者の機能を同



（a） 外部制御型ユーザインタフェース管理システム



（b） 内部制御型ユーザインタフェース管理システム

図 4 ユーザインタフェース管理システムの構造
Fig. 4 Structures of user interface management systems.

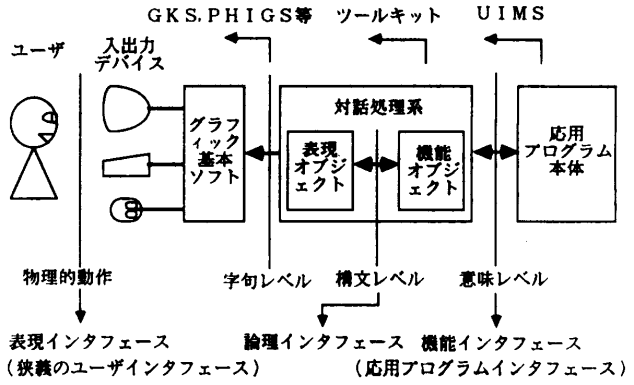


図 5 ユーザインタフェースソフトウェア構成
Fig. 5 Structure of user interface software.

時に実現する構成とすることが望ましいが，単純に両者を混合したのではソフトウェア構成が複雑になり，応用プログラムとユーザインタフェースの分離がすっきりしなくなる．したがってこの問題を自然な形で解決するシステム構成が求められる．

3.4 ユーザインタフェース構築環境の課題

従来の UIMS ではユーザインタフェースの定義を宣言的言語で記述し，これをコンパイルすることにより実行形式のプログラムを得ていた[1),3),7)]．しかし，詳細な文法を覚える必要があることや，画面上の表示状態や UI 部品の配置は実行させてみないとわからないという問題があった．そこで本システムでは，画面上で部品を組み立てる要領で対話的にユーザインタフェースを構築できるツールを提供することとした．

ユーザインタフェース構築効率の向上には，定義に要する作業量が少ないことはもちろん，試行錯誤的な定義が可能なことが望まれる．このためにはメニューやアイコンなど，対話に用いられる表示物（これらを UI 部品と呼ぶ）を動的に生成したり，その属性や表示位置，大きさ等を自由に変更できる機能が必要である．

4. 対話処理系の構成

4.1 対話処理系の位置付け

UIMS の対話処理系はユーザと応用プログラム本体の中間に位置する．ユーザとのインタフェース（狭義のユーザインタフェース）はユーザの物理的動作の対象であるメニューやアイコン等の UI 部品の集合であり，これを表現インタフェースと呼ぶ．一方応用プログラム本体とのインタフェースは応用プログラム内の手続きの呼出しや，データの入出力等，応用プログラム側から見た対話機能を表し，これを機能インタフェースと呼ぶ．そして両者間の変換処理を行うのが UIMS の対話処理系である．

前述のように階層モデルはユーザと応用プログラムの対話の表現に優れており，階層モデルに基づいて対話処理系を構成することは対話管理上有力であると考えられる．すなわち，図 5 に示すようにユーザインタフェースソフトウェアの構成を字句レベル，構文レベル，意味レベルの処理に分割し，階層化する．機能インタフェースを意味レベル，グラフィック基本ソフトとのインタフェースを字句レベルととらえる．さらに対話処理系を構文解析レベルと意味解析レベルに分割し，両者間の構文レベルのインタフェースを論理インタフェースと呼ぶこととする．

階層モデルを用いると UIMS の機能を理解しやすい．すなわち図 5 に示すように，GKS や PHIGS のようなグラフィック・サブルーチン・パッケージは字句レベルの処理まで，ユーザインタフェース用のツールキットは構文レベルまでの処理を提供するのに対し，UIMS は意味レベルまでをサポートする点で優れていると考えられる．

4.2 対話処理系の基本構成

対話処理系は 3.3 節で述べた課題を満足する構成でなければならない．そこで本システムではオブジェクト指向プログラミングを採用した．

これにより，①モジュール性が高いため応用プログラムとユーザインタフェースの分離を実現しやすい，②UI 部品をオブジェクトとして自然に表現できる，③オブジェクトの変更や修正は容易で，これによりユーザインタフェースを簡単に変更できる等，前述の要求を満足できる．また従来から言われているように

オブジェクト指向は直接操作によるユーザインタフェースの実現にも向いている．

ところでユーザインタフェースでは機能的には同じ対話処理を異なる形式で表現することが多い．例えば応用プログラム中の特定の手続きを起動する方法には文字列コマンド入力，メニュー選択，アイコン指定などの方法がある．階層モデルに当てはめれば構文レベルでは異なっても意味レベルでは同じ対話が存在することを表している．

そこで本システムではUI部品を表現形式に関する（構文レベル）処理を行うオブジェクトと，対話の抽象的機能に関する（意味レベル）処理を行うオブジェクトに分割し，前者のみの変更で表現インタフェースを変えられるようにした．前者を表現オブジェクト，後者を機能オブジェクトと呼ぶ．

表現オブジェクトにはメニュー，アイコン，文字列入出力用部品，メータや温度計などの数値入出力用部品，グラフ等があり，応用分野やユーザの好みなどによりさまざまな種類が考えられる．一方機能オブジェクトは手続き呼出しオブジェクト，文字列入出力オブジェクト，数値入出力オブジェクト，集合選択オブジェクト等であり，基本的には応用プログラムの記述言語やデータ型によって決定される．

上記の分離により，対話処理系は図5のように表現オブジェクト群と機能オブジェクト群に分割できる．一般に両者のインタフェース，すなわち論理インタフェースは単純なデータのやりとりとなる．例えばメニュー選択による応用プログラム内の手続き呼出しはメニューオブジェクトと手続き呼出しオブジェクトで実現するが，メニュー項目の番号が論理インタフェースの情報となる．

4.3 対話処理系の全体構成

対話処理系の構成を**図6**に示す．対話処理系は応用プログラムとグラフィック基本ソフト（ウィンドウマネジャやグラフィックパッケージ）の間に位置し，表現オブジェクトと機能オブジェクトが基本的な構成要素である．ウィンドウオブジェクトは表現オブジェクトの一種であるが，後述するように表現オブジェクトの管理上重要な働きをする．イベント管理オブジェクトは入力イベントを監視し，イベント形式の対話を支援する．履歴管理オブジェクトは取り消しや再実行等の対話処理を実現するため，履歴情報を記憶しておくオブジェクトである．

対話処理系を構成するオブジェクトのクラス階層を**図7**に示す．最上位にはオブジェクトの基本的機能を実現する基本オブジェクトのクラスが位置し，その下位に各クラスが存在する．クラスの継承機能を利用することにより効率的なプログラミングが可能であり，特に多種多様な表現オブジェクトや機能オブジェクトの実現に大きく役立っている．

本システムはエンジニアリングワークステーション

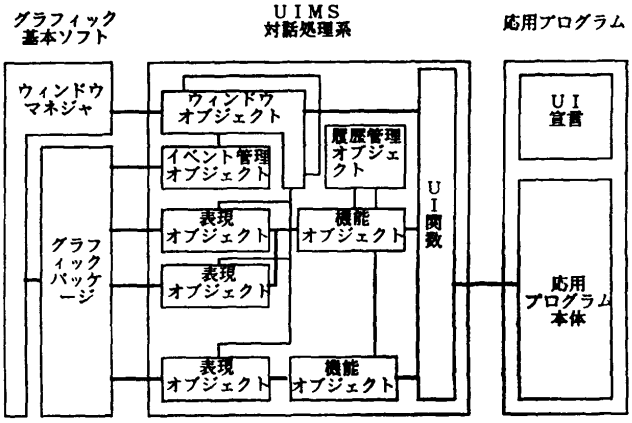


図6 対話処理系の構成
Fig. 6 Architecture of the dialog manager.

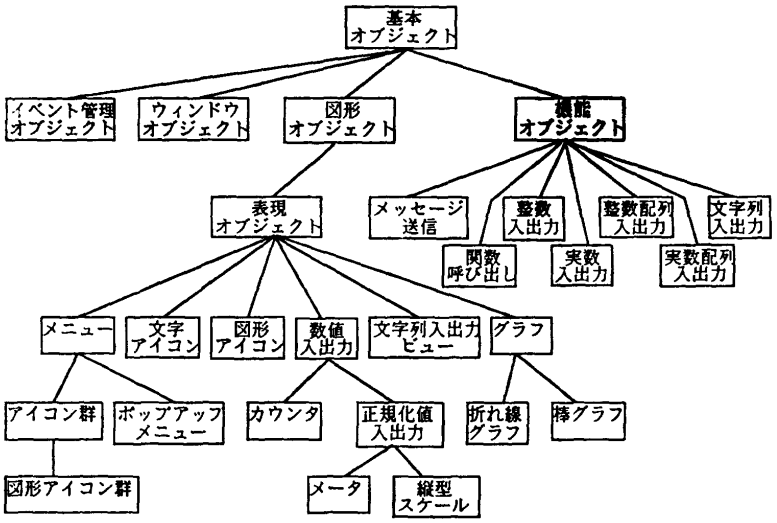


図7 クラスの階層
Fig. 7 Class hierarchy.

ES-330 上に試作した．記述言語はCを採用し，メッセージパッシングやクラスの継承，インスタンスの動的生成などの機能を含む，オブジェクト指向の枠組みをC言語上に実現している．

5. 対話管理方式

5.1 対話処理

まず対話処理の動作について説明する．表現オブジェクトと機能オブジェクトはメッセージを交換しながら処理を進めるが，両者間には特定の制御の方向は存在せず双方向の処理が可能である．したがってイベント形式とリクエスト形式の対話を同じように扱え，外部制御型，内部制御型両者の UIMS の機能を同一構成で実現できる．また，アクティブ・バリュー[13]の機能を導入して，応用プログラム中のデータを書き替えた時に自動的に機能オブジェクトを起動することにより，モニタ形式の対話を実現する．このように本システムは3種の対話形式を同一構成のもとで実現している．

次にデータの表現形式に関する課題についてであるが，さまざまな種類の表現オブジェクトを用意してデータの多様な表現を可能とするとともに，それらの交換で表現形態の変更を容易に行えるようにした．

5.2 画面管理

本システムでは表現オブジェクトを表示上の従属関係に従って階層化して管理する．この方式を図8を用いて説明する．同図上方の CRT 画面の最上部のウィンドウには四つのアイコンと文字列表示用の UI 部品であるテキストビュー，固定メニュー，図形表示用の UI 部品であるグラフィックビュー，自由図形（車の絵），ポップアップメニューが表示されているが，これらの従属関係の階層構造を同図下方に示す．ウィンドウオブジェクトが最上位にあり，その下に四つのアイコンをもつアイコン群とテキストビューが従属している．テキストビューの下位には固定メニューとグラフィックビューが，さらにグラフィックビューの下位に自由図形とポップアップメニューがある．

一般に従属関係において下位の表現オブジェクトほど表示の優先順位は高く，したがって上に表示される．また下位のオブジェクトは上位のオブジェクトの表示領域内に含まれるのが普通である．

階層構成による画面管理では上位のオブジェクトに対する処理は下位のオブジェクトにも影響する．例えばテキストビューを画面上から消去すれば固定メニューやグラフィックビュー等も消去される．移動，拡大，縮小等の処理も同様である．なお下位のオブジェクトに対する処理が上位へ及ぶことはない．

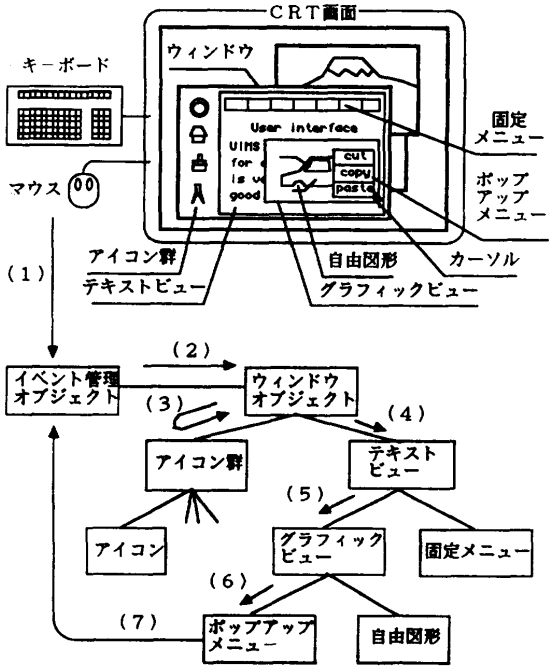


図 8 画面管理とイベント管理
Fig. 8 Display control and event management.

以上のように表現オブジェクトを従属関係に従って階層化することで，統一的な画面管理が可能になる．

5.3 イベント管理

イベント形式の対話ではユーザからのイベントが予測できないため，その管理方式が大きな課題である．本システムではイベント管理オブジェクトと表現オブジェクトの従属関係の階層構造を利用することによりイベント管理を実現する．イベントとしてはキーボードやマウス等からの入力が代表的である．イベント管理について図8を用いて説明する．

本システムではイベント管理オブジェクトがイベントを監視している．イベント情報はボタンやキーの種類と座標から成る．今，マウスのボタンがクリックされたとすると，イベント管理オブジェクトはウィンドウオブジェクトにイベント情報を引数としたイベントメッセージを送信し，イベント処理を依頼する．

ここでイベントメッセージを受信したオブジェクト（ウィンドウオブジェクトまたは表現オブジェクト）の動作を説明する．まず自分の下位のオブジェクトにイベントメッセージを送信し，すべてのオブジェクトがそのイベントと無関係と返答した時に自分に関する

イベントかどうかを判定し，そうであればイベント処理を実行し，そうでなければその旨を上位のオブジェクトに返答する．

入力イベントがその表現オブジェクトに関するものかどうかはイベント情報により判定する．一般にはイベント座標が表現オブジェクトの表示領域内に含まれるかどうかで判定するが，ポップアップメニューのように表示位置がイベント座標に依存する場合は所定の感応領域内かどうかで決める．下位のオブジェクトから判定を行っていくのは，下位のオブジェクトほど表示の優先順位が高いからである．

図8の例では，まず，マウスからのイベント(1)をイベント管理オブジェクトが取り込み，ウィンドウオブジェクトに報告する(2)．ウィンドウオブジェクトはまず，アイコン群に対してイベントメッセージを送信する(3)が，座標（カーソル位置）がアイコン群の外側なので，無関係と判定される．次にテキストビュー(4)，グラフィックビュー(5)，さらにポップアップメニューにイベントメッセージが送られる(6)．ポップアップメニューは自分が処理するイベントと判定し画面上にポップアップ表示し，項目選択のイベント(7)に従ってメニュー選択処理を実行する．

以上のように表現オブジェクトの階層構造を利用することにより統一的なイベント管理を可能とした．この方法は表現オブジェクトの追加や削除を行ってもイベント管理処理の修正は不要という特徴をもつ．

6. ユーザインタフェース構築環境

6.1 全体構成

応用プログラム開発をも含めたユーザインタフェース構築環境の全体構成を図9に示す．

ユーザインタフェース構築用のツールはUIエディタ，グラフィックエディタ，ビットマップエディタおよびそれらを管理するUIMSマネジャより成る．UIエディタは応用プログラムのユーザインタフェースを画面上で対話的に定義するもので，その結果はUI定義データファイルに出力される．グラフィックエディタとビットマップエディタはユーザが自由な図形や画像を作成するためのツールである．その結果は図形データファイルに出力され，UIエディタから読み込んで，UI部品の形状データとして利用できる．

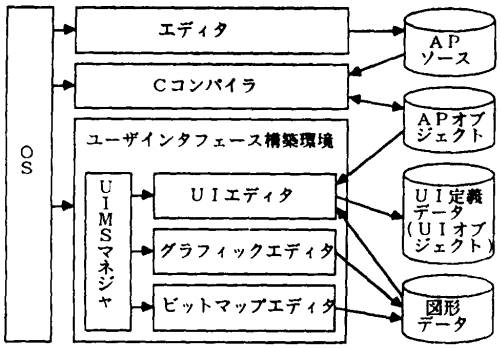


図9 全体環境
Fig. 9 Programming environment.

6.2 応用プログラムの記述方法

ここで応用プログラムの記述方法について簡単に述べる．本システムでは応用プログラム本体の作成後にユーザインタフェースを付加するという手順を想定している．基本的に応用プログラムでは具体的なユーザインタフェースを意識する必要はない．しかし，両者を接続するための最低限の情報が必要である．例えばイベント形式で呼び出す応用プログラム内の関数名やモニタ形式で値を表示する変数名などの情報を宣言する必要がある．これをUI宣言と呼ぶ．

またリクエスト形式の対話では入出力に使用するUI部品を指定できなければならないが，応用プログラム側ではその詳細は考えたくない．そこで応用プログラムではUI部品を抽象化し，抽象デバイスとして扱えるようにした．そして対話処理系に入出力を要求するためのインタフェースとしてUI関数を提供する．

応用プログラムの記述例を図10に示す．記述言語はCである．UI宣言では“func_call( )”の形で呼び出す関数名を，“ad_double( )”の形で実数型デー

UI宣言

```
func_call(start, stop, end);        /* 関数宣言 */
ad_double(speed_meter);             /* 抽象デバイス宣言 */
av_int(number);                     /* モニター宣言 */
```

応用プログラム本体

```
start()
{
    int number;
    double X;
     . . .
    ui_output(speed_meter, X);      /* リクエスト形式の出力 */
    . . .                           /* モニター形式 */
    number = 128;
}

    . . . . . .
```

図10 応用プログラム記述例
Fig. 10 An example of application program.

タを入出力する抽象デバイス名を宣言している．応用プログラム本体では UI 関数 “ui_output( )” を用いてリクエスト形式の出力要求を行っている．このように記述された応用プログラムの具体的なユーザインタフェースは UI エディタを用いて定義する．

6.3 UI エディタ

UI エディタは画面上で部品を組み立てる要領で，対話的にユーザインタフェースを構築するためのツールである．UI エディタは応用プログラム側の UI 宣言を読み出し，抽象デバイスに対応する機能オブジェクトを生成した後，ユーザとの対話により表現オブジェクトの生成や属性定義を行う．

UI エディタによるユーザインタフェースの定義方法を，さきほどの応用プログラムの中の抽象デバイス “speed_meter” を例にとり，**図 11** を用いて説明する．最初に UI エディタは抽象デバイスの型（実数入出力）に使用可能な UI 部品（表現オブジェクト）の種類をメニュー表示しユーザに選択を促す(1)．ここではメータを選ぶ．次にメータのタイトル(2)，最大値(3)，最小値(4)，色(5)の属性定義を行う．次に表現オブジェクトの階層における上位のオブジェクト（親）を指定する(6)．ここではウィンドウを選択したが，任意の UI 部品をピック指定できる．最後に表示領域をマウスにより指定する(7)．こうして(8)のような UI 部品が得られた．

UI エディタを使用すれば以上のような手順を繰り返して，UI 部品の集合であるユーザインタフェースを簡単に構築できる．

7. 評価と今後の展開

7.1 対話処理系の評価と今後の課題

試作した UIMS の対話処理系について処理速度と機能の面から評価を行った．

まず処理速度についてはオブジェクト指向や階層構成を利用したイベント管理を採用したことから若干の不安があったが，そのオーバヘッドはほとんど見られない．処理速度は入出力ハードウェアやグラフィック基本ソフトによって決まると考えてよい．

次に機能面であるが，あまり複雑でないユーザインタフェースについては十分目的を達したと言える．しかし高度なユーザインタフェースの実現にはいくつかの問題が残されている．

まず，複雑な処理を実現するには機能オブジェクトの動作をユーザが自由に定義できるようにする必要があり，さらにその動作を動的に変更できることが望ましい．また本システムでは対話がひとつひとつシーケ

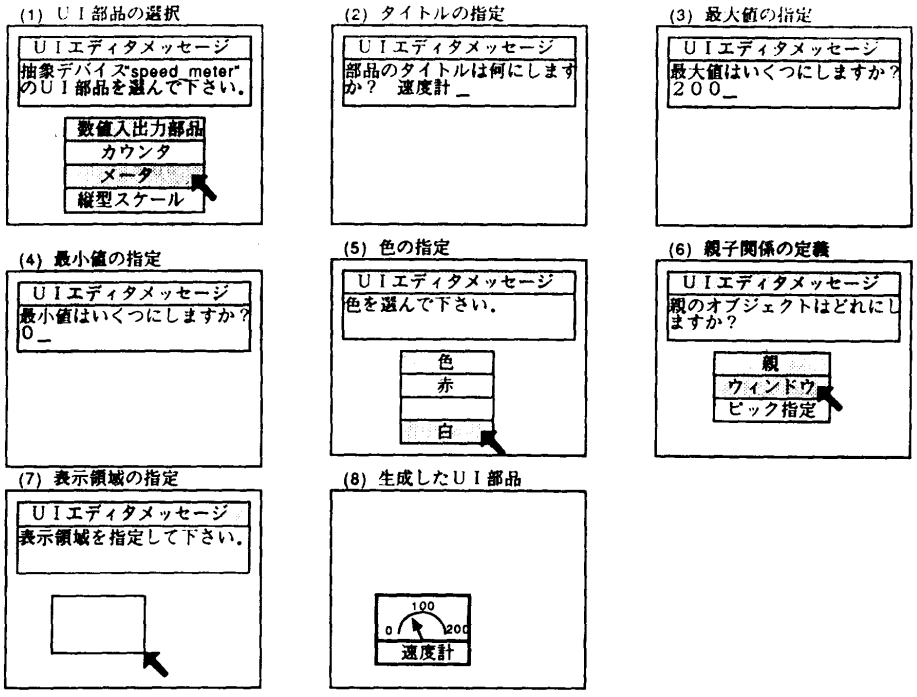


図 11 UI 部品の対話的定義例
Fig. 11 An example of interactive definition of user interface object.

ンシャルに実行されることを想定していたが，一つの対話を保留状態にしておいて別の対話を実行したいことがある．そのためにはリクエスト形式の入力とイベント形式の対話を同じ形式で管理する必要がある．

7.2 構築環境の評価と今後の課題

定量的な評価は難しいが，本システムを用いて実際にユーザインタフェースを対話的に定義する方法は，従来のプログラミングによる記述と比較して格段に効率が向上する．この点では対話的な定義は非常に有力である．

しかし実用化にあたっては環境をもっと整備する必要がある．まず，本システムではユーザインタフェースのデバッグを支援するための環境をまだ提供していないが，応用プログラムとは独立に実行可能なユーザインタフェースのデバッガが不可欠である．

また，本システムでは応用プログラム作成後にユーザインタフェースを定義するという形式を前提としているが，実際にはユーザインタフェースと応用プログラム本体は並列に設計されることが多い．このようなプログラム作成過程を支援できる環境とする必要がある．しかし，それにはユーザインタフェースのみでなく，応用プログラム本体の開発をも含んだ，統合的なプログラミング環境とする必要がある．

7.3 今後の展開

ここでは本システムをさまざまな計算機システムに展開する方法について述べる．

本システムでは応用プログラムに対話処理系をリンクして一つのロードモジュールを形成するが，マルチプロセス環境では複数の応用プログラムプロセスに対し一つの UIMS プロセスで対処する[14]ことが考えられる．しかし UIMS には応用プログラムに密着した部分があることやプロセス間の通信量を少なくする必要があることから，単純に両者を別プロセスにすることはできない．

ところが表現オブジェクトと機能オブジェクトの間の論理インタフェースで分離することによりこれらの問題を解決できる．というのは論理インタフェースはやりとりされるデータ量が最も少ないうえ，表現オブジェクトの処理は入出力装置に依存するが応用プログラムには依存せず，機能オブジェクトの処理は応用プログラムには依存するが入出力装置に依存しないからである．

また，メインフレームの端末にパーソナルコンピュータやワークステーションを使用したシステムに適用する場合には，応答性のよいことはもちろん，端末の種類が変わっても応用プログラムの変更なしで使用可能としたい．それにはユーザとの直接的な対話を行う部分を端末が，その他の部分をメインフレームが実行すればよく，前と同様に論理インタフェースで分離すればよい．

同様のことは複数のワークステーションによる分散処理環境についても言える．この場合を図 12 に示す．各ワークステーションには複数の応用プログラムプロセスと一つの UIM (ユーザインタフェースマネジャ) プロセスが存在する．この形態はウィンドウシステムのクライアント・サーバ・モデル[15]の発展形態と言うこともできる．

このように階層モデルに基づいた UIMS はさまざまな形態のシステムに柔軟に対応できる．

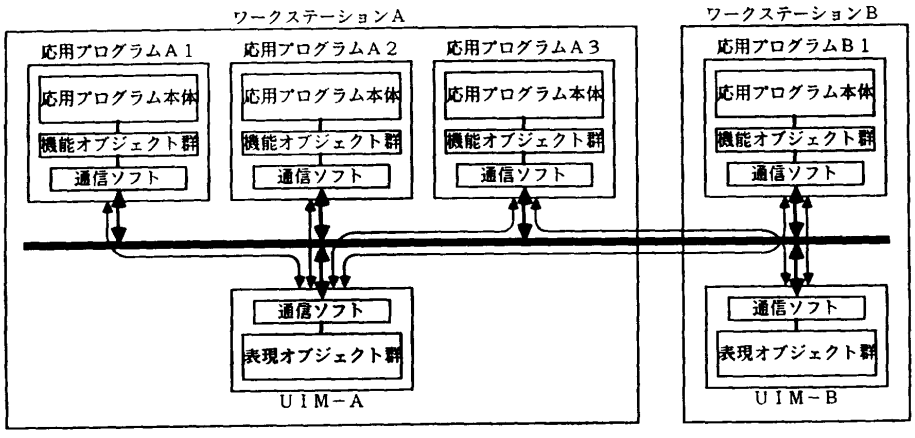


図 12 分散型 UIMS
Fig. 12 Distributed user interface management system.

8. むすび

ユーザと応用プログラムの対話を表現する階層モデルに基づいたユーザインタフェース管理システムを提案し，実際に試作，評価を行った．本システムは実行時に対話処理を行う対話処理系と，ユーザインタフェース構築環境から成る．

対話処理系はオブジェクト指向により実現したが，ユーザとの直接の対話に関する処理をする表現オブジェクトと応用プログラムと密接に関係する処理をする機能オブジェクトの組合せを基本としている．ユーザインタフェースの処理をこの二つのオブジェクトに分離したことにより，修正，変更を容易にするとともに，さまざまなシステムに適用可能とした．また表現オブジェクトを木構造に階層化することにより，統一的な画面管理，イベント管理を実現した．

さらに画面上で対話的にユーザインタフェースを構築できるUIエディタを有する構築環境を実現することにより，ユーザインタフェース構築効率を大幅に向上することを可能とした．

現在，本システムを知識処理やCADに適用し，実用化を図っている．

謝辞 本研究の機会を与えていただくとともに，有益な御意見をいただいた，(株)日立製作所日立研究所の平沢宏太郎部長，増田郁朗部長，板東忠秋部長，ソフトウェア工場の原田千秋主任技師，高橋修主任技師，おおみか工場の大島啓二主任技師に感謝する．

参考文献

1) Kasik, D. J.: A User Interface Management System, *Computer Graphics*, Vol. 16, No. 3, pp. 99-106 (1982).
2) Thomas, J. J., Hmamlin, G. et al.: Graphical Input Interaction Technique (GIIT), *Computer Graphics*, Vol. 17, No. 1, pp. 5-30 (1983).
3) Buxton, W. A., Lamb, M. R., Sherman, D. and Smith, K. C.: Towards a Comprehensive User Interface Management System, *Computer Graphics*, Vol. 17, No. 3, pp. 35-42 (1983).
4) Pfaff, G. E. (ed.): *User Interface Management Systems*, Springer-Verlag, Berlin (1985).
5) Green, M.: The University of Alberta User Interface Management System, *Computer Graphics*, Vol. 19, No. 3, pp. 205-213 (1985).
6) Hayes, P. J., Szekely, P, A. and Lerner, R. A.: Design Alternatives for User Interface Management Systems Based on Experience with Cousin, *CHI '85 Proceedings*, pp. 169-175 (1985).
7) Schulert, A. J., Rogers, G. T. and Hamilton, J. A.: ADM—A Dialog Manager, *CHI '85 Proceedings*, pp. 177-183 (1985).
8) Card, S. K., Moran, T. P. and Newell, A.: *The Psycology of Human-Computer Interaction*, Lawrence Erlbaum Associates, Hillsdale, New Jersey (1983).
9) Shneiderman, B.: Direct Manipulation: A Step beyond Programming Languages, *IEEE Computer*, Vol. 16, No. 8, pp. 57-69 (1983).
10) Bullinger, H. J. and Faehnrich, K. P.: Symbiotic Man-Computer Interfaces and the User Assistant Concept, in Salvendy, G. (ed.), *Human-Computer Interaction*, pp. 17-26, Elsevier Science Publishers B. V., New York (1984).
11) Sisson, N.: Dialog Management Reference Model, *SIGCHI Bulletin*, Vol. 18, No. 2, pp. 34-35 (1986).
12) Foley, J. D. and van Dam, A.: *Fundamentals of Interactive Computer Graphics*, Addison-Wesley, Reading, Massachusetts (1982).
13) Stefik, M. J., Boblow, D. G. and Kahn, K. M.: Integrating Access-Oriented Programming into a Multiparadigm Environment, *IEEE Software*, Vol. 3, No. 1, pp. 10-18 (1986).
14) Endele, G.: Report on the Interface of the UIMS to the Application, in Pfaff, G. E. (ed.), *User Interface Management Systems*, pp. 21-29, Springer-Verlag, Berlin (1985).
15) Gosling, J.: Partitioning of Function in Window Systems, in Hopgood, F. R. A. et al. (eds.), *Methodology of Window Management*, pp. 101-106, Springer-Verlag, New York (1986).

(昭和63年5月9日受付)
(平成元年2月14日採録)

横山 孝典(正会員)

1959年生．1981年東北大学工学部通信工学科卒業．1983年同大学院工学研究科電気及通信工学専攻修士課程修了．同年(株)日立製作所入社．日立研究所にて図形認識，ユーザインタフェースの研究開発に従事．1987年より(財)新世代コンピュータ技術開発機構に出向．知識表現の研究に従事．電子情報通信学会会員．

谷　正之（正会員）

1956 年生．1980 年東京工業大学工学部電気・電子工学科卒業．1982 年同大学院物理情報工学専攻課程修了．同年(株)日立製作所日立研究所に入社．文書処理，連想記憶，オブジェクト指向プログラミング環境の研究・開発を経て，ユーザインタフェース関連ソフトウェアの研究・開発に従事．現在 MIT メディアラボにて客員研究員としてユーザインタフェースの研究中．

荒井　俊史（正会員）

1964 年生．1986 年東京工業大学理学部情報科学科卒業．現在，(株)日立製作所日立研究所にてユーザインタフェース関連ソフトウェアの研究・開発に従事．プログラミング言語，プログラミング環境，OS 一般に興味をもつ．

谷藤　真也（正会員）

昭和 22 年生．昭和 48 年早稲田大学理工学研究科修了．同年(株)日立製作所日立研究所入社．鉄鋼プロセスの計算機制御，知識工学応用システムの研究に従事．現在はエンジニアリングワークステーションのソフトウェアおよびユーザインタフェースの研究を担当．計測自動制御学会，電気学会などの会員．