文書処理とヒューマンインタフェース 23 － 1
（１９８９．３．８）

プロダクション・システムに基づいた ユーザインタフェース管理システム

宮崎 一哉，福岡 久雄，辻 順一郎，坂下 善彦
三菱電機（株） 情報電子研究所

梗概

ワークステーションの普及に伴い、ウィンドウやメニュー等を用いた対話的で視覚的なユーザインタフェースに対する要求が高まっている。このようなユーザインタフェースをを効率良く開発するためのツールとして、かつ優れたユーザインタフェースを追及するためのテスト・ベッドとして、我々はユーザインタフェース管理システム（ＵＩＭＳ）を開発している。ＵＩＭＳは、ユーザインタフェースに係わる部分をアプリケーション・プログラムの他の部分から切り離して集中的に管理するシステムである。本稿では、今回開発した、プロダクション・システムに基づいた対話制御部を持つＵＩＭＳの構成と機能について述べる。

Production-System-Based User Interface Management System

Kazuya MIYAZAKI, Hisao FUKUOKA, Junichiro TSUJI, Yoshihiko SAKASHITA
Information Systems and Electronics Development Laboratory
Mitsubishi Electric Corporation
5-1-1 Ofuna, Kamakura City
247 Japan

Today's popularity of workstations makes request for the graceful user interfaces which are visual and interactive greater. We are researching on user interface management systems as tools for higher productivity of user interface part of application programs and as test beds for the research on more graceful user interfaces. User interface management systems, in a centralized way, control the user interface part separated from the other program parts. This paper describes the structure and functions of the production-system-based user interface management system that we have developed.

1．はじめに

ワークステーション等の普及に伴い、計算機システムの利用者層が拡大し、計算機の非専門家が計算機システムを利用する機会が増大した。それに従い、ウィンドウ、メニュー、マウスなどを駆使した視覚的で対話的なユーザインタフェースに対する要求が高まってきているが[1]、「優れた」ユーザインタフェースを実現するための具体的な手法が明確でないためユーザインタフェース部の開発は試行錯誤的になる傾向にあり、このことがアプリケーション全体の開発効率に大きな悪影響を与えている。

従来のアプリケーション開発の経験から、アプリケーション・プログラムは、アプリケーションの本来の機能を果たす部分とユーザインタフェースに係わる処理を行う部分とに分離でき、かつユーザインタフェース部の多くの機能が広範囲のアプリケーションに対して共通に利用され得ることが指摘されている。このような考えに基づき、ユーザインタフェース部のみを独立に実現し、集中的に処理しようというシステムの研究開発が進められている。このようなシステムをユーザインタフェース管理システム（ＵＩＭＳ）と呼ぶ[2]。

ＵＩＭＳは、ユーザインタフェースの設計、実装および評価を総合的に支援するシステムであり、ユーザとアプリケーション本体部の間に位置して、両者の間の対話制御を行う。ＵＩＭＳを用いることにより、より優れたユーザインタフェースに向けての改良がアプリケーション本体に何ら手を加えることなく行うことができるようになる。また同一のアプリケーションに対して複数のユーザインタフェースを用意することも容易となる。

我々は、M.Green 等によって提案されたユーザインタフェース・モデル（ Seeheimモデル）に従ったＵＩＭＳを研究している[3]。今回、プロダクション・システムをベースとした対話制御部（後述）を持つＵＩＭＳの実験システムを当社のエンジニアリング・ワークステーションME1000のウィンドウ・システム上に構築した。本稿では、以降、Seeheimモデルの簡単な紹介に続いて、本実験システム概略と今後の課題について述べる。

2．Ｓｅｅｈｅｉｍモデル

Seeheimモデル[2]は、ユーザインタフェースの論理モデルの１つである。このモデルでは、ユーザインタフェースをPresentation Component、Dialogue Control Component、Application Interface Modelの３つの構成要素に分割して考えている（図１）。

Presentation Componentは、ユーザインタフェースとして用いられるすべての入出力デバイスの管理を行う。さらに、すべての対話技法および表示フォーマットは、この要素で定義される。Presentation Componentは、入力デバイスからの生データをユーザインタフェース中の他の構成要素に対して、その構成要素が要求する形式に抽象化した後、受け渡す。また、他の構成要素から渡された抽象的な出力データを解釈し、それらを出力デバイス上の物理的表現形式に変換する。これらの抽象的なデータ表現は、データのタイプと値から構成される。この一かたまりの情報はトークンと呼ばれる。

Dialogue Control Componentは、Presentation Componentから順次送られてくる入力ト

ークンを解釈してユーザの意図を決定する。この要素はユーザインタフェースの制御ロジックを含み、ユーザの意図を決定した段階でApplication Interface Modelを用いて適当なアプリケーション・ルーチンを呼出す。同様に、Application Interface Modelから送られてくる出力トークンは、この構成要素によって解釈され、Presentation Componentへの出力トークンに変換される。

Application Interface Modelは、アプリケーションの機能の表現であり、アプリケーションが管理する主要なデータ構造の記述と、ユーザインタフェースにより呼び出されるアプリケーション・ルーチンを含んでいる。さらに、ユーザインタフェースとアプリケーションとの間の通信形態も含んでいる。この通信形態には、外部制御、内部制御および混合制御の３種類がある。

３．全体の構成

今回作成した実験システムは、以下のような特徴を持っている。

◎ユーザインタフェース設計ツールによる、ウィンドウ、メニュー、フォームの対話的定義

◎宣言的記述による対話処理定義

◎外部制御方式に基づく対話制御

◎マルチウィンドウを利用したユーザインタフェース環境

全体のプログラム構成を図２に示す。

表示制御部はME1000既存ウィンドウ・システムとリンクされ、ウィンドウ、メニュー、フォームなどの定義データを含む表示定義部と併せてSeeheimモデルのPresentation Componentに相当する機能を担う。

対話制御部は、対話定義部に記述された対話の宣言的記述にしたがって、ユーザとアプリケーション間の対話制御を行うプログラム・モジュールであり、SeeheimモデルのDialogue Control Componentに相当する機能を担う。

FM1、…、FMnはアプリケーション本来の機能を果たす機能モジュール群である。各機能モジュールは、対話制御部からの呼び出しによりその機能を果し、対話制御部にリターンする（外部制御方式）。

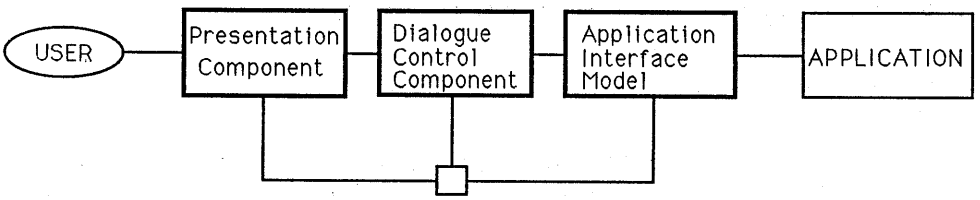

図１　Ｓｅｅｈｅｉｍモデル

4．表示制御部

表示制御部は、ME1000上の既存のウィンドウ・システムに次にあげる機能を追加することによりPresentation Componentに相当する機能を果している。

(1) ウィンドウ、メニューおよびフォームの定義および表示処理

ユーザとの対話手段として、各種グラフィック入力が可能な通常ウィンドウ、文字列入力専用ウィンドウ、ポップアップ・メニューおよびフォームを用意し、これらの定義を対話制御部とは独立に行う。また、この定義データに基づいた表示処理を行う。

(2) トークンの生成

ユーザからの入力データを抽象的な形式（トークン）に変換して対話制御部に渡す。

(3) 入力処理

既存ウィンドウシステムからの入力方式を対話制御部にとって望ましい形態に変換する。

4．1　ウィンドウ、メニューおよびフォームの定義および表示処理

既存のウィンドウ・システムのアプリケーション・インタフェースでは、例えばウィンドウを表示する場合、ウィンドウのサイズ、位置、色等の表示属性を定める定義データをパラメータとしてウィンドウ生成コマンドを与える。優れたユーザインタフェース実現のためには、ウィンドウの表示属性を操作の流れや他のウィンドウとの関係等から決定する必要があり、ウィンドウの表示と表示属性の修正を繰り返すことが望ましい。この修正にあたり、従来の方法ではアプリケーション・プログラム本体の変更、再コンパイル、リンクが必要となり、その繰り返しのサイクルは大きくなり、効率が悪い。

そこで本システムでは、ウィンドウ、メニューおよびフォームの定義データを表示定義部としてファイル内に格納しておき、それらを名前（論理デバイス名）を用いて指示することにより表示、活性化および消去できるような機能を用意した。

また、ウィンドウ、メニューおよびフォームの表示属性や論理デバイス名の定義を対話的かつグラフィカルに行ない、それを表示定義部に格納するために、ユーザインタフェース設計ツール[4]が用意されている。

この方式により、ウィンドウ、メニューおよびフォームの表示属性の変更は、対話制御部や機能モジュールとは独立に、表示制御部だけで行うことが可能となる。また、ユーザインタフェース設計ツールを用いることにより、ウィンドウ等を実際に表示して確認を行いながら設計できるため、表示属性の決定が効率良く行える。

4．2　トークンの生成

表示制御部では、ユーザからの入力を対話制御部に渡すに際して、既存のウィンドウ・システムから得られる生データを対話制御部で扱いやすい論理的な形式（トークン形式）に変換する。

トークンは次の形式を取る。

［論理デバイス名、トークンの型、トークンの値］

論理デバイス名は、表示定義部に定義された個々のウィンドウ、メニュー等の識別名で

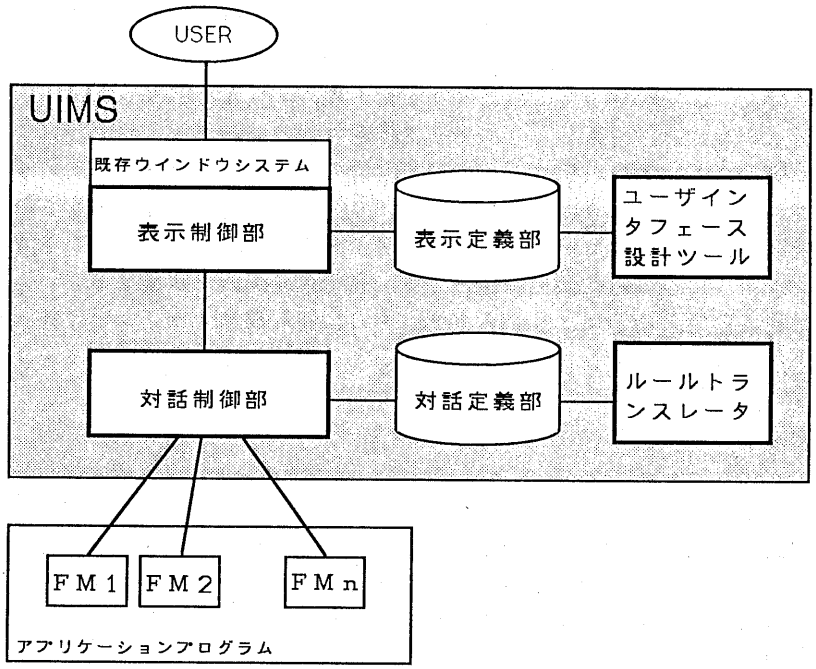

図２　全体構成

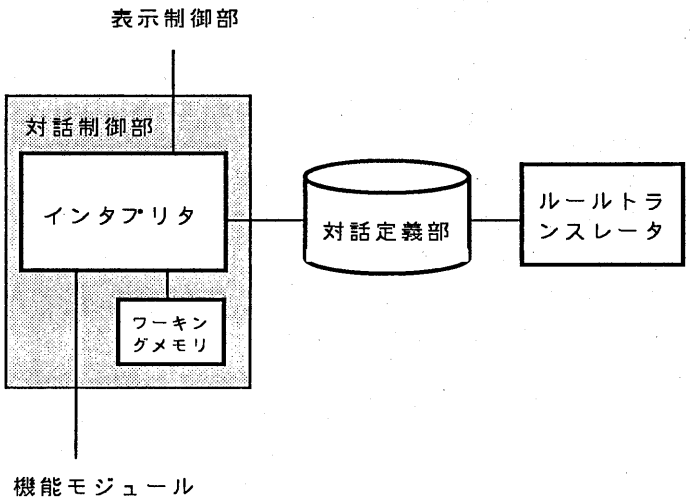

図３　対話制御部の構成

あり、対話定義に用いることのできる論理的な名前である。表示定義部に定義され、表示制御部がウィンドウ等の実体へのポインタとの対応を管理する。

トークンの型は、論理デバイスの種別（ウィンドウ、メニュー、フォーム）、データの型（文字／ポインティング情報など）によって規定されている。

トークンの値は、トークンの型によって形式が異る。ウィンドウ内のポインティング情報やキー入力、およびフォームの文字列入力用エリアへの入力は、予め規定された形式の値を表示制御部自身が生成するが、メニュー・アイテムやウィンドウのソフトウェア・ボタン等にはプログラマが自由に値を設定できる（表示定義部内に定義する）。

このような処理を行うことにより、入力データの詳細な構造やウィンドウ等へのポインタの管理を意識せずにユーザとアプリケーションとの間の対話設計を行うことができる。

4．3　入力処理

既存ウィンドウ・システムではユーザからの入力データをウィンドウ毎に存在するキューに振分けて格納しているために、複数のウィンドウにまたがるユーザ入力を発生時刻順に読むことはできない。一方、ＵＩＭＳではすべてのユーザ入力を発生時刻順に得て対話を制御する必要があるため、ウィンドウ毎に入力専用プロセスを貼り付けて発生時刻順の読み込みを可能にしている[5]。

5．対話制御部

対話制御部は、表示制御部によって提供された抽象度の高いインタフェースを利用してプログラマが記述したユーザとアプリケーションの間の対話定義を処理するプログラム・モジュールである。

対話制御部はプロダクション・システムをベースとしている。プロダクション・システムを対話制御部のベースとして採用した理由は次のとおりである。

◎記述能力が高い
◎対話を宣言的に記述でき、記述間の独立性が高い
◎解釈系の作成が容易である
◎知的処理の導入が容易である

対話制御部の構成を図３に示す。対話制御部は、ワーキング・メモリ、対話定義部（ルールベース）、インタプリタで構成される。さらに、プログラマが記述するための記述ルールをシステムによる実行形式に変換するルール・トランスレータがある。

以下の節で、各構成要素の概要を紹介する。

5．1　ワーキング・メモリ

ワーキング・メモリは、対話制御部が対話を進めるにあたって、その時点時点で認識しておく必要のある世界の状況に関する情報を保持するモジュールである。対話が進むにつれてその内容は更新される。具体的には、次のような情報を保持する。

◎アプリケーションの状態名

アプリケーションの状態は、プログラマがアプリケーションを設計するときに想定してルール中に記述するものであり、この状態名の書換え等の管理はすべてルールによって行う。これを考慮することによって、ルールをモジュール化して捉えることができ、対話の設計がしやすくなり、インタプリタによるルールの解釈の際に検索範囲を限定して実行速度を向上することができる。

◎ユーザからの入力トークン

ユーザからの入力は、表示制御部によって所定の形式のトークンに変換されてワーキング・メモリに格納される。

◎機能モジュールからのリターン値

ルールの実行によって起動された機能モジュールがリターン値を持つ場合は、それがワーキング・メモリに格納される。

◎システムの状態を表す各種フラグ／スロット

各種フラグ／スロットは、ユーザによるコマンドやパラメータなどの設定の状態を一時的に保持しておくためなどに用いる。

５．２　対話定義部（ルール・ベース）

対話定義部は、対話制御部にたいして与えられる対話を定義するルール形式のデータ（単にルールと呼ぶ）の集まりである。ルール・トランスレータにより実行形式に変換されたルールが構造的に（後述）格納されている。

各ルールは、そのルールを特定するためのルール名、実行するルールを選択するときに参照する優先度、条件部、そして実行部から構成され、

「もし条件部が成立するならば実行部の処理を行え」

という意味を持つ。

条件部の成立／不成立はワーキング・メモリの内容との照合の成功／不成功によって決まる。条件部にはワーキング・メモリ内で扱う値を直接記述する他、それらの値にたいする許容範囲を記述するという間接的な記述を行うことができる。

実行部には、表示制御部にたいする入出力の要求、機能モジュールにたいする起動メッセージ、ワーキング・メモリの内容にたいするアクセス・メソッド（各種フラグのオン／オフ、スロットへの値の代入）等を記述する。

また、各ルールにたいしては、優先度を付随しており、これに基づいて実行すべきルールの選択が行われる。

５．３　インタプリタ

インタプリタは、ユーザとシステムとの間の対話を制御するＵＩＭＳの中枢部である。

一般の素朴なプロダクション・システムの処理アルゴリズムは、まず、ワーキング・メモリの内容と各ルールの条件部との間で照合を行い、次に照合の成功したルールの中から実行すべきルールをある戦略に基づいて一つ選択し（競合解消）、選択されたルールの実行部を処理するという３つの動作を繰り返すものである。

今回作成した実験システムでは、まず、アプリケーションの状態名により照合を試みるルールの範囲を限定し、さらに優先度の高い順に照合を行って、照合に成功したルールをすぐに実行することにより、競合の解消のためのステップを踏まずに済ませ、実行効率を向上させている。

5．4　ルール・トランスレータ

ルールには、アプリケーション・プログラマが記述するためのルール（記述用ルール）とインタプリタが解釈実行するためのルール（実行用ルール）の２種類が存在する。記述用ルールはプログラマにかかる負担が最も少なくなるように、なるべくシンプルで判り易いことが要求され、実行用ルールはインタプリタにかかる負担がなるべく小さくなるような形であることが要求される。従って両者の形は異なったものとなる。ルール・トランスレータは、記述用ルールを実行用ルールに変換し、対話定義部に格納するツールである。ルール・トランスレータを起動し、プログラマによって与えられたルールをシステムによる実行形式に変換すると、ルールに記述したアプリケーションの状態名によるルールの構造化、および優先度によるルールの並べ替えが同時に行われ、対話定義部に格納される。

6．おわりに

現在、本システムを利用して電子メール・システムを構築し、評価を行っている。現時点で、今後の課題として検討すべき項目として、次の項目があがっている。

◎新たな入力技法のついかを容易にする機構
◎表示オブジェクトを中心とした対話定義
◎ダイレクト・マニピュレーションのサポート

ダイレクト・マニピュレーションに関しては、Seeheimモデルの不適合性が言われている。現在のモデルに適した領域の見極め、ダイレクト・マニピュレーションに適した枠組の追求が必要である。

参考文献

[1] 宮崎：ユーザインタフェースに対する要求，情処　夏のシンポジウム，1986.
[2] G.E.Pfaff：User Interface Management Systems, Springer-Verlag, 1985.
[3] 坂下他：ユーザインタフェース・システム－構成概要－，情処第３５回全国大会，1987.
[4] 福岡他：ユーザインタフェース設計ツール－画面設計支援－，情処研究会　文書処理方式とヒューマンインタフェース17-3，1988.
[5] 福岡他：ユーザインタフェース・システム－プレゼンテーション・コンポーネント－，情処第３５回全国大会，1987.