言語モデルおよびMVC構造に基づくユーザインタフェース管理システム—GUIDMAS†

今宮淳美†† 関村勉†††*

本論文は，言語モデルおよびオブジェクト指向システムSmalltalkのMVC構造に基づくユーザインタフェース管理システム(UIMS), GUIDMASの設計およびSmalltalkでの実現について述べる．従来，ユーザインタフェース作成に広く利用されているモデルは，インタフェースをユーザとコンピュータ間の対話とみる言語モデルである．すなわち，対話を語彙，構文，および意味レベルに構造化してある．しかし今日では，グラフィックス技術の発展にともない，図形と直接対話する直接操作による対話が注目されている．すなわち，視覚的に表現されたオブジェクトに操作（コマンド）を提示することでユーザが仕事を容易に実行でき，結果も直ちに見ることができる作業環境をユーザに提供するUIMSが必要である．よりよいユーザインタフェースは言語処理とオブジェクト指向両方の要素を持つべきとの観点から，これら2つの考え方を融合するUIMS, GUIDMASを設計，および実現した．GUIDMAS設計の主眼は，構成要素をいかに分離して，それら要素間の制御および通信をどう達成するか，およびオブジェクトをどう構造化すべきか(MVC構造がUIMS設計に適しているか)である．現在，GUIDMASの第1版がTektronix 4404上で動作している．

1. はじめに

現代の人間とコンピュータとの対話（相互作用）は，従来のコマンド言語だけでなく，メニュー選択，直接操作などを組み合わせた形式でなされて複雑化している[1]．このため，ユーザインタフェースに関する学際的研究が様々な観点からなされている．これら研究分野の中で対話を扱うソフトウェアシステムは，ユーザインタフェース管理システム（UIMS）と呼ばれている．1986年，ACM/SIGGRAPHのワークショップ[2]ではそれまでの議論をまとめて「UIMSとは，ユーザインタフェースの設計，およびユーザと応用システムとの相互作用を管理するソフトウェアシステムである」と定義づけられた．したがって，UIMSに関係する2つのユーザタイプが存在する．すなわち，ユーザインタフェースソフトウェア設計者とエンドユーザである．このためUIMSは，インタフェース設計者に対して複雑な対話を制御する対話型システムの開発ツール群を提供する．一方，エンドユーザに対しては，高度な対話を可能とするインタフェースを提供する．したがって，UIMSの設計と実現には，システムのモデル化，またはシステムアーキテクチャに関する研究が重要である．

これまでユーザインタフェース作成に広く研究および利用されているUIMSのモデル（またはアーキテクチャ）は，インタフェースをユーザとコンピュータ間の対話とみる言語モデルである[3]．すなわち，対話を語彙，構文，および意味レベルに分割して各レベルでの問題点に焦点をあてようという枠組みでSeeheimモデル[4]として提案されている（**図1**参照）．プレゼンテーション部には，入出力デバイス，スクリーン配置，対話技法（ツール）があり，この要素は語彙レベルとみることができる．対話制御部は，ユーザとコンピュータ（または応用システム）間の対話を扱う．すなわち，ユーザが用いる対話構造を扱うことから，この要素は構文レベルとみることができる．応用インタフェースモデル部は，ユーザインタフェースとそれ以外のプログラム間のインタフェースを定義する．この要素は，応用プログラムの手続き呼び出しを扱ったり，フィードバック，デフォルト値および誤り検査のような意味的操作に関する情報を与えたりその応用特有のヘルプ情報を提供することから，意味レベルとみることができる．

従来，Seeheimモデルでの対話制御部を対象として，オートマトン，拡張遷移ネットワーク（ATN），または言語トランスレータ作成ツール（たとえば，yaccとlex）を利用してのシステム設計および実現に関する多くの研究がある[3]．しかし今日では，グラフィックス技術の発展にともない直接操作による対話が重要になりつつある[1]．すなわち，視覚的に表現されたオブジェクトに操作を提示することでユーザが仕

† GUIDMAS—A UIMS Based on the Linguistic Model and the MVC Structure by ATSUMI IMAMIYA and TSUTOMU SEKIMURA (Department of Electrical Engineering and Computer Science, Faculty of Engineering, Yamanashi University).

†† 山梨大学工学部電子情報工学科
*† 現在 富士通(株)

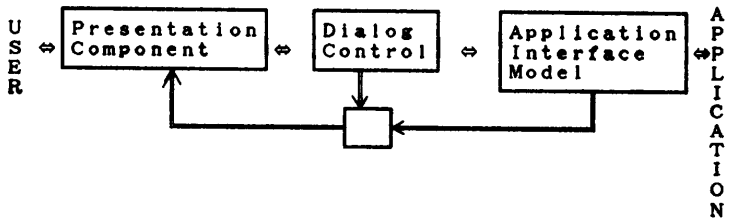


図 1 Seeheim モデル
Fig. 1 Seeheim model.

事を容易に実行でき，結果も直ちに見ることができる作業環境をユーザに提供するインタフェースである．直接操作では，ユーザはシステム全体とではなく，注目する個々のオブジェクトと対話すると考える．すなわち，UIMS をオブジェクト指向システムとみる．この考えから，UIMS の重要な課題である対話の並行処理および複数同時対話[6] (multi-threaded dialogues) 研究への見通しも良くなると期待できる．

オブジェクト指向による UIMS の第一の利点として，オブジェクトの継承の性質が既存の対話技法を修正し，利用しやすくすることが期待できる．しかし，プログラミングの枠組みを与えるだけでは使いやすいユーザインタフェース開発は望めない．したがってオブジェクト指向プログラミングが，いかにユーザインタフェース管理システムに適しているか，ユーザインタフェースを扱うのにオブジェクトをどう構造化すべきか，たとえば本論文の観点のように Smalltalk[5] の MVC 構造が適しているかどうか，またはオブジェクト指向プログラミングが UIMS モデルにどのように影響するかなどが研究課題である．

筆者らは，よりよいユーザインタフェースは，言語とオブジェクト指向の両方の要素を持つべきと考えている．本論文では，言語モデルでの 3 要素，およびオブジェクト指向システムのひとつである Smalltalk の MVC パラダイムの 3 要素を融合する 3 つのオブジェクトから成るユーザインタフェース管理システム GUIDMAS の設計および作成について述べる．この GUIDMAS 設計の主眼は，構成要素をいかに分離し，それらの要素間の制御および通信をどのように達成するかにある．

2 章では GUIDMAS 設計の考え方，構造および各要素の機能を説明する．3 章では GUIDMAS が提供するインタフェース開発用システム A-kit について述べ，4 章でインタフェース作成過程の例を示す．

2. ユーザインタフェース管理システム (UIMS) —GUIDMAS

1 章で述べたように，これからのユーザインタフェースソフトウェア研究の重要な課題のひとつは，(オブジェクト) 直接操作対話，および複数同時対話を提供する UIMS についてである．

本論文では，それらの対話を提供するオブジェクト指向 UIMS を設計・試作してアーキテクチャ，すなわち構造要素，要素間の同期と情報伝達 (データ，および制御) について考察する．

Hill は複数同時対話のためのイベント型対話記述システムを作成した[6]．Jacob は直接操作対話を記述するのに遷移図を用いている[9]．しかし，それらの研究は，UIMS 構造については言及していない．オブジェクト指向システムにおけるユーザインタフェースアーキテクチャを，Smalltalk-80 のソースコード中に見ることができる．それは，モデル・ビュー・コントローラ (MVC) 構造と呼ばれる．Smalltalk のユーザインタフェース (ツール群) はこの考え方に基づいて作られているが，MVC 構造自体についてはまだ詳しい研究報告がなされていない．

さらに，複数同時対話および/または (オブジェクト) 直接操作対話を提供する UIMS のアーキテクチャがいかにあるべきかについての詳しい研究は現在のところ見あたらない．

本論文では，複数同時対話を提供する UIMS のアーキテクチャとして MVC 構造が言語モデルと共存できることを GUIDMAS の設計・作成を通して示す．さらに，GUIDMAS をオブジェクト指向システムとすることで直接操作対話に対応できる UIMS の一構成法を提案している．

2.1 MVC 構造と GUIDMAS

Smalltalk-80 には，ウィンドウシステム利用の対話型プログラムを開発するときの設計指針となる，モデル・ビュー・コントローラ (MVC) 構造と呼ばれる基本的枠組みがある．モデル，ビュー，コントローラの 3 つのオブジェクトが相互にメッセージを送り，全体で 1 つの Smalltalk 上の応用システムを構成する．これは，ビューとコントローラが協調してモデルのためのユーザインタフェースを実現するという考え方で

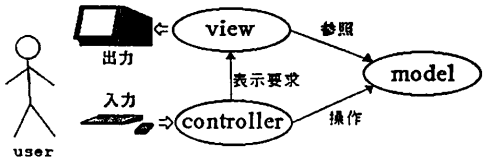


図 2 MVC 構造
Fig. 2 MVC structure.

ある（**図 2** 参照）.

モデルは，プログラムの中心的存在で，他のオブジェクト（またはユーザ）からの操作対象になるオブジェクトである．情報の保持変更，および対象のオブジェクトを自然に記述することがモデル・オブジェクトの役割である．ビューは，対象のモデル，自分の表示範囲（ウィンドウ），およびモデルの表示方法を知っているオブジェクトであり，“display” メッセージを受け取ることでモデルを参照し，ウィンドウ上にモデルの情報を表示する．コントローラは，ユーザからの指示をキーボードやマウスからの入力として受け取り解釈してモデルまたはビューにメッセージを分配したり，入力に応じたフィードバックをビューに行わせる．

2.2 GUIDMAS の構成

GUIDMAS は，対話ビュー，対話フローマネージャ，アプリケーションインタフェースの 3 つのモジュールからなる（**図 3** 参照）．この 3 モジュールは，Seeheim モデルの 3 要素と対応する．すなわち，対話ビューとプレゼンテーション部（語彙レベル処理），対話フローマネージャと対話制御部（構文レベル処理），アプリケーションインタフェースと応用インタフェース部（意味レベル処理）が対応する．また，GUIDMAS は，Smalltalk の MVC 構造にも対応する．GUIDMAS と Smalltalk の MVC 構造との間では，対話ビューとビュー，対話フローマネージャとコントローラ，アプリケーションインタフェースとモデルが対応する．この 3 つのモジュールでは，他のモジュールへメッセージを送ることにより情報を伝達し，そのモジュールを起動して制御を渡す．

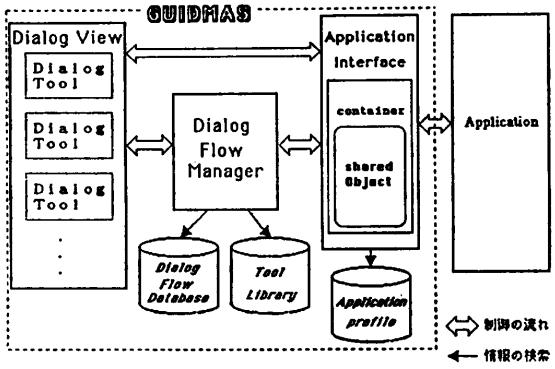


図 3 GUIDMAS の構造
Fig. 3 The organization of the GUIDMAS.

UIMS 設計における重要な問題のひとつは，応用プログラムと UIMS 間の通信である．すなわち，応用の意味情報をインタフェース設計にいかに組み込むかである．従来から，UIMS と応用との分離という考え方は議論されている[3]．しかし，この論文で対象とする直接操作や高度な相互作用システムにおいては，どこでどのようにその分離をするかは非常に難しい．2.2.2 項のアプリケーションインタフェースの構造と機能の説明でこの問題に対する一方法を示す．

2.2.1 対話ツールと MVC 構造

GUIDMAS は，Smalltalk-80 が提供する対話ツールを利用するとともに，ポップアップメニューを除く対話ツールを MVC 構造に基づいて作成する．GUIDMAS で利用する Smalltalk-80 の対話ツールを生成するクラスと MVC との対応を**表 1** に示す．テキストメニューとアイコンメニューのモデル（オブジェクト）を生成するために GUIDMAS で新たにク

表 1 対話ツールと MVC 構造
Table 1 Dialog tools v. s. the MVC structure.

Dialog tools	Model	View	Controller
Binary Choice	BinaryChoice	BinaryChoiceView	BinaryChoiceController
Text Menu	TextMenu	SelectionInListView	SelectionInListController
Icon Menu	FormMenu	FormMenuView	FormMenuController
Text Workspace	Workspace	WorkspaceView	StringHolderController
Graphics Workspace	Form	FormView	FormEditor
Text Input Tool	FillInTheBlank	FillInTheBlankView	FillInTheBlankController
PopUp Menu	PopUpMenu	——	——
GUIDMAS	ApplicationInterface	DialogView	DialogFlowManager

ラス TextMenu とクラス FormMenu を作成してある．Smalltalk-80 が提供するアイコンメニューとしてのビットエディタ (Bit Editor) では，生成するインスタンスのメニュー内容，アイコンの位置などを変更できない．GUIDMAS では，任意の項目（アイコン）を任意の位置に設定できるように Smalltalk-80 の標準クラスを書き換え，アイコンメニューのコントローラとビューを生成するクラスを持つようにしてある．

2.2.2 アプリケーションインタフェース

GUIDMAS と応用を分離するために，GUIDMAS から起動する応用プログラムへのメッセージセレクタのフォーマット，および応用プログラムの起動条件をアプリケーションプロファイルとして持つ（**表 2** 参照）．

アプリケーションプロファイルは 2 要素を持つクラス Array のインスタンスである．2 つの要素は，ともにクラス String のインスタンスであり，第一要素が応用の起動条件式，第二要素がアプリケーションを起動するメッセージ式である．起動条件式の評価結果が真のとき，第二要素の文字列を実行することで応用を起動する．文字列でのメッセージ式は，クラス Compiler に “evaluate:” メッセージを送ることで実行できる．これは Lisp の eval 機能に相当する．

アプリケーションインタフェースは，アプリケーションプロファイルを用いて次のように応用の起動を行う．

```
applicationProfile do: [: anArray|
            “アプリケーションプロファイルの各
            要素を引数 anArray で受け取る”
  conditionString ← anArray at: 1.
            “anArray の第一要素を condition-
            String とする”
  executeString ← anArray at: 2.
            “anArray の第二要素を execute-
            String とする”
(Compiler evaluate: conditionString) ifTrue:
  [Compiler evaluate: executeString].
            “conditionString の評価結果が真な
            らば，executeString を実行する”
]
```

表 2 アプリケーションプロファイル記述
Table 2 The description of an application profile.

```
起動条件式
    (MainMenu==#PLACE)
      & (SymbolMenu==#CHAIR)
応用プログラム起動式
    TinyRoomLayout place: #CHAIR
```

アプリケーションインタフェースは，GUIDMAS と応用プログラム間で受け渡しする情報を保持するコンテナオブジェクトを持つ．コンテナは，応用プログラムに伝えるユーザからの入力情報を持つ．アプリケーションインタフェースは，コンテナの持つ情報と起動条件を照合し，アプリケーションプロファイルからメッセージレシーバとなるべき応用プログラムを決定しその応用プログラムを起動する．

コンテナは使用している対話ツールのモデルを保持する共有オブジェクト（インスタンス変数 sharedObject）を持つ．応用プログラムは，コンテナを受け取ることでそのときに使用している対話ツールの情報を得て，モデルの情報を書き換えることで意味レベルのフィードバックがなされる．さらにコンテナは，アプリケーションプロファイルを検索するためのインスタンス変数 activeToolNames と activeToolStatus を持つ．activeToolNamesは，現在使用しているツール名を保持する配列である．activeToolStatus は，現在使用しているツールの状態（メニューの選択項目，文字入力ツールで入力された文字など）を保持する配列である．この activeToolNames と activeToolStatus の配列の大きさは等しく，その内容は一対一対応している．すなわち，activeToolNames の i 番目の名前を持つツールの状態は，activeToolStatus の i 番目の要素として保持される．また，activeToolStatus の要素のデータ型は，使用するツールによって異なるため，テキストメニューならば選択項目のシンボル，文字入力ツールならば文字列，ポップアップメニューならば選択項目の順番を表す数値である．さらに activeToolStatus は，オブジェクト指向の動的結合を利用し任意のデータ型のオブジェクトを保持する（シンボル，文字列，数値，その他）．

activeToolNames と activeToolStatus の役割は，使用する対話ツールの情報を素早く取り出すことである．このためコンテナを応用プログラムに送り，応用プログラムは sharedObject よりも簡単にツールの持つ情報を取り出すことができる．また，アプリケーションプロファイルを検索するには，activeToolNames と activeToolStatus を起動条件式の引数として渡し，起動条件式の各変数に割り当てる．表 2 に示す起動条件式の主メニューでは ‘PLACE’ が選択さ

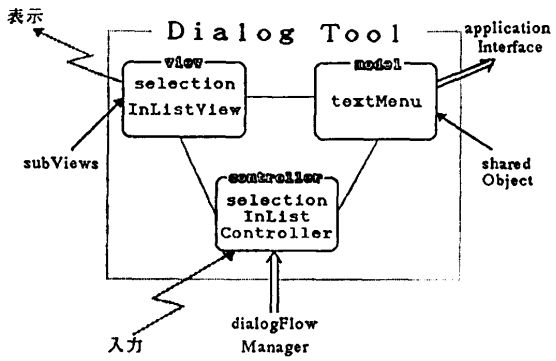


図 4 対話ツールの MVC 構造
Fig. 4 The MVC structure of the dialog tool.

れ，シンボルメニューで 'CHAIR' が選択されるときは起動式が評価され，応用プログラムであるオブジェクト 'TinyRoomLayout' へメッセージ 'place: #CHAIR' が送られる．

2.2.3 対話ビュー

対話ビューはすべての対話ツールのビューに対するスーパビューであり，GUIDMAS のビュー階層構造により対話ツール間の同期を制御する．複数のビューを使用する場合，あるビューが他のビューのスーパビューかサブビューであれば，各ビュー相互に制御を移動できる．現時点で使用している対話ツールのすべてのビューを制御するスーパビューの下に階層構造を持つ．

対話ツールは，MVC 構造に基づき3つのオブジェクトから成る（**図4**参照）．ツールのモデルはツールライブラリ内に保管される．ツールが使用されるとき，対話フローデータベースはツールのモデルをコンテナの sharedObject に登録する．またモデルに対応するビューのクラスからインスタンスを生成し，モデルとモデルへのアクセス方法を知らせ，さらに対話ビューの subViews に登録しサブビューを作る．ツールのコントローラは，サブビューが作られた時点で自動的に生成される．その後，ツール内にマウスカーソルが入ったならば，対話フローマネージャは（対話ビューツールのビューを通して）ツールのコントローラを起動する．コントローラはマウスやキーボードの入力を受け取り，ビューの書き換えとモデルの情報を更新する．モデルは，情報が更新されたことをアプリケーションインタフェースに伝えて応用の起動を決定させる．

表3におけるクラス DialogView は対話ビューを生成するとともに，スーパクラスの StandardSystemView からビューとしての主な機能を継承する．

表 3 GUIDMAS のクラス階層
Table 3 Hierarchy of the GUIDMAS classes.

```
Object
 ├View[1]
 │ ├BinaryChoiceView[2]
 │ ├ListView[1]
 │ │ └SelectionInListView[2]
 │ ├StringHolderView[1]
 │ │ ├WorkspaceView[2]
 │ │ └FillInTheBlankView[2]
 │ ├FormMenuView[2]
 │ ├FormView[2]
 │ └StandardSystemView[1]
 │   └DialogView[3]
 ├Controller[1]
 │ ├BinaryChoiceController[2]
 │ ├FormMenuController[2]
 │ └MouseMenuController[1]
 │   ├FormEditor[2]
 │   ├ScrollController[1]
 │   │ ├ListController[1]
 │   │ │ └SelectionInListController[2]
 │   │ └ParagraphEditor[1]
 │   │   └StringHolderController[2]
 │   │     └FillInTheBlankController[2]
 │   └StandardSystemController[1]
 │     └DialogFlowManager[3]
 ├Collection[1]
 │ ├Set[1]
 │ │ └Dictionary[1]
 │ └SequenceableCollection[1]
 │   └ArrayedCollection[1]
 │     ├Array[1]
 │     └String[1]
 ├PopUpMenu[2]
 ├BinaryChoice[2]
 ├DisplayObject[1]
 │ └DisplayMedium[1]
 │   └Form[2]
 ├StringHolder[1]
 │ ├FillInTheBlank[2]
 │ └Workspace[2]
 ├MenuTool[3]
 │ ├TextMenu[3]
 │ └FormMenu[3]
 ├ApplicationInterface[3]
 ├Container[3]
 └Guidmas[3]
```

[1] smalltalk-80 で既定義のクラス
[2] 対話ツールのインスタンス作成に利用するクラス
[3] GUIDMAS クラス

2.2.4 対話フローマネージャ

対話フローマネージャは，対話ツールの生成，対話ビューの起動および制御を行う．このためツールライブラリには，各対話ツールの情報を保持するオブジェクト，すなわち対話ツールのオブジェクト（モデル）がある．

ユーザインタフェースの設計者は，ユーザとコンピュータ間の対話の流れを対話フローとして記述する．対話フローは，応用システムの動作をユーザとコンピュータ間の入出力で表す対話の流れの情報であり，GUIDMAS では Scene ネットワークでこれを表現する．従来，対話の流れは，メニューの木またはネット

ワーク構造で表現されている[1]．メニューネットワークは，ひとつのメニュー項目を選択することにより，他のメニューへ遷移する対話の表現方法である．したがって，ユーザは複数メニューを同時に使うことはできない．GUIDMAS では，複数の対話ツールを同時に使う場面（Scene）を扱う．そのために各場面で複数の対話ツールを並列に動作させ，ユーザ入力により他 Scene への遷移を表す Scene ネットワークで対話フローを記述することで対話制御を行う．対話フローマネージャは，現在の Scene に必要な対話ツールを対話ビューのサブビューとして生成する．ユーザ入力が遷移条件を満たす場合には，他 Scene へ遷移し，その Scene 内で必要とする対話ツールを生成する．GUIDMAS では，ユーザインタフェース設計者が記述する対話フローを対話フローデータベースに格納するので，応用プログラムが対話や応用プログラム自体の流れを管理する必要はない（4章に例を示す）．

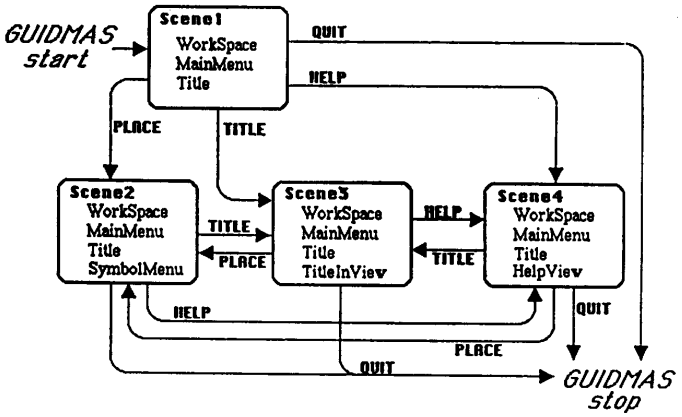


図 5 Scene ネットワークの構造例
Fig. 5 Typical structure of a Scene network.

Scene ネットワークの例を**図 5** に示す．GUIDMAS が起動すると最初に対話フローの Scene1 を検索する．この Scene1 は，WorkSpace, MainMenu, Title の3つの対話ツールを使用する．対話フローマネージャは，ツールライブラリからこの3つの対話ツールのモデルを検索し，それらをアプリケーションインタフェースの sharedObject に登録する．さらに各対話ツールのビューに相当するオブジェクトを生成し，それらを対話ビュー内にサブビューとして登録する．Scene 内で使用する対話ツールと，他 Scene への遷移条件の設定後，各サブビューのコントローラに制御を移してユーザの入力を待つ．

クラス DialogFlowManager は，GUIDMAS 全体のコントローラである対話フローマネージャを生成する．この DialogFlowManager は，スーパクラスの StandardSystemController から，サブビューの制御，およびマウスカーソルを領域内に含んだときの処理など主なビュー機能を継承する（表3参照）．

2.3 GUIDMAS の動作

GUIDMAS のシステム全体の動作を詳細に説明する．GUIDMAS を構成するオブジェクトと通信の様子を**図 6** に示す．

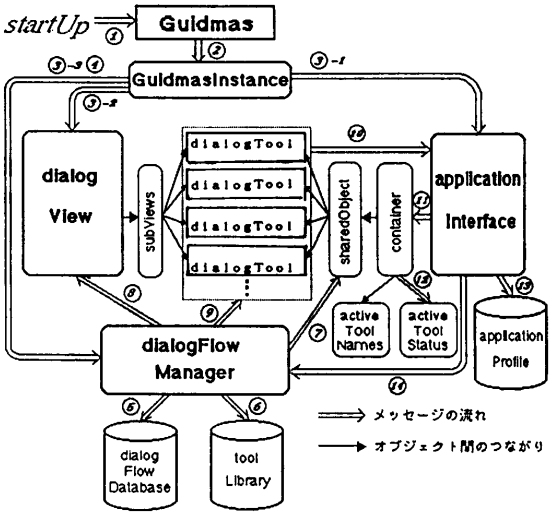


図 6 GUIDMAS の動作
Fig. 6 Mechanism of the GUIDMAS.

① ユーザまたは応用からクラス Guidmas にメッセージを送り，対話管理システム GUIDMAS を起動する．

② クラス Guidmas は大域変数 GuidmasInstance を生成し，それに制御を移す．

③ GuidmasInstance はクラス DialogView，クラス ApplicationInterface，クラス DialogFlowManager から各インスタンスを生成し，対話ビュー，アプリケーションインタフェース，対話フローマネージャの各モジュールを作る．また，ユーザが指定するアプリケーションプロファイル，対話フローデータベース，およびツールライブラリの各データベースを使用するモジュールに知らせる．

④ 対話フローマネージャを起動する．

⑤ 対話フローマネージャは，currentScene をキーと

して対話フローデータベースから Scene の検索をする.

例：検索結果→#(#(#MainMenu#WorkSpace #Title) 遷移条件式)

⑥ ⑤の検索結果の第一要素をキーとしてツールライブラリからユーザインタフェースツールの検索をする.

検索結果→#(ツールのモデル 対話ビュー内の位置)

⑦ ⑥の検索結果の第一要素を sharedObject に登録する.

⑧ ツールのビューに対応するオブジェクトを生成し，対話ビューの subViews に登録する. このときの対話ビュー内におけるサブビューの位置には⑥の検索結果の第二要素を指定する.

⑨ ⑥〜⑧を⑤の検索結果の第一要素のすべてのツール名に対して行う. 終了したならば，対話フローマネージャのコントローラとしての機能を起動し，マウスカーソルを含むサブビューのコントローラに制御を移す.

⑩ ツールに対してユーザの入力があるならば，ツールのコントローラはモデルの情報を書き換える. ツールのモデルは，このメッセージを受け取ることにより入力が発生したことを知り，アプリケーションインタフェースを起動する.

⑪ アプリケーションインタフェースは，コンテナへ activeToolStatus の更新，および引渡しを要求する.

⑫ コンテナは activeToolStatus を更新し，その情報をアプリケーションインタフェースへ渡す.

⑬ アプリケーションインタフェースは更新された activeToolStatus を参照しながら，アプリケーションプロファイルの起動条件式を評価する. 実行条件が整っていればアプリケーション起動式を評価する.

⑭ アプリケーション起動後，アプリケーションインタフェースは対話フローマネージャに再起動の要求を出す. このとき，対話フローマネージャは遷移条件式を評価し，currentScene が変更されたら⑤へ戻る. 変更されなかったら，再びサブビューのコントローラに制御を渡す.

3. ユーザインタフェース開発支援機能 —A-kit—

GUIDMAS では，ユーザインタフェース設計者のためのユーザインタフェース開発支援機能 A-kit を提供する. 現在，A-kit には次に示すエディタを用意してある：

A-kit Tools Library

- Text Menu Editor
- Icon Editor
- Dialog Flow Editor
- Application Profile Editor

Icon Editor は，Smalltalk の Bit Editor をもとに設計試作されている. さらに，アイコンメニューの Model として FormMenu を作成した.

A-kit は，GUIDMAS のツールライブラリ，対話フローデータベース，アプリケーションプロファイル，および GUIDMAS から起動される対話型応用システムであり，これを使うことで，ユーザインタフェースの設計と利用が容易となる. すなわちユーザインタフェース設計者は，対話ツール作成時に各ツールの属性を指定するためのメッセージセレクタを記憶したり，ツールライブラリへの登録方法を覚える必要がない.

A-kit を起動すると，A-kit の各種対話ツールが対話ビュー内に表示される. ユーザインタフェース設計者は，これらのツールを使い既定義のクラスからインスタンスを生成し，それらを視覚的に編集してツールの属性を詳細に指示し，対話ツールライブラリにそのインスタンスを登録する. この A-kit の既定義メニューツールなどを用いて新しく対話ツールを編集することでユーザインタフェース設計者は各オブジェクトのメッセージセレクタやプロトコルを記憶しておく必要がない. さらに A-kit を使って Smalltalk-80 の既定義クラスのユーザインタフェースのインスタンス生成，および属性の変更ができる.

4. インタフェース作成過程

これまで述べてきた GUIDMAS と A-kit 機能動作を，簡易部屋配置プログラム[8] 作成過程で説明する.

GUIDMAS での対話型システムの開発には次に示す 4 つの作業が必要である.

（1） 対話ツール作成,

（2） 対話フロー指定,

（3） アプリケーションプロファイル作成，
（4） GUIDMAS から起動する応用プログラム作成，

以下にユーザインタフェース作成過程（1）～（3）を説明する．

（1） 対話ツール作成

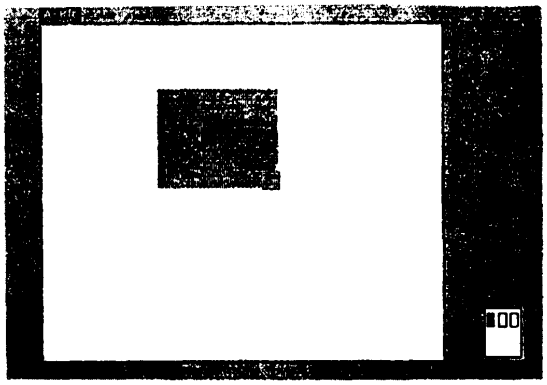
はじめに，簡易部屋配置で使用する主メニューを A-kit の Text Menu Editor を用いて作成する．この Editor は，はじめにインタフェース設計者にメニューの位置と大きさを指定するよう求める（**図7**（a））．領域指定が終ると A-kit はコンテナの sharedObject 内部に TextMenu のインスタンスを生成し，これにより空項目のメニューを表示する．インタフェース設計者がこのメニュー領域内にマウスカーソルを移動させマウスの中ボタンを押すと，メニューの属性指定ポップアップメニューを表示する（図7（b））．インタフェース設計者は，このポップアップメニューを用いて属性を指定してテキストメニューを作成する．次にこのメニューにツール名 'MainMenu' と名付けてツールライブラリに登録する．

（2） 対話フロー指定

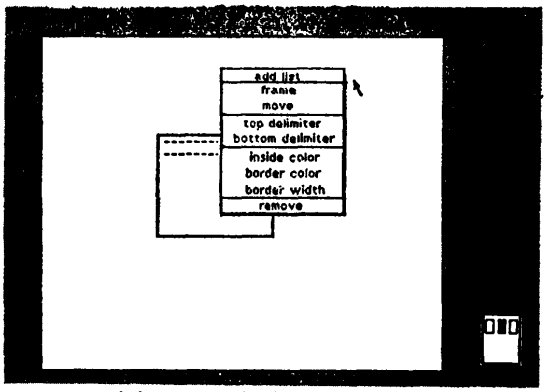
対話ツール作成後，A-kit の Dialog Flow Editor を用いて，Scene ネットワークを作成する．簡易部屋配置の Scene ネットワーク（図5）を作成するには，最初の Scene（Scene1）で使用するツールが MainMenu, WorkSpace, Title の 3 つであることを指定し，さらに他の Scene への遷移条件式を Smalltalk-80 のメッセージ式で記述する（**図8**）．これらの式では変数名がツール名であるので，Dialog Flow Editor を用いて対話フローデータベースに Scene ネットワークを登録するとき，ツール名を適当な一時変数に置き換えてツールの入力情報をその一時変数に設定する式を自動的に加える．図8に示す遷移条件式では，主メニューの選択結果が 'PLACE' ならば Scene2 へ 'TITLE' ならば Scene3 へ遷移するよう指定している．この例では単一ツールの入力結果による遷移を示しているが，インタフェース設計者は複数の対話ツールへの入力による遷移条件式も記述できる．

（3） アプリケーションプロファイル作成

アプリケーションプロファイルを起動条件式と応用プログラム起動式の2つの式で記述する．起動条件式は，ツールが指定の状態にあるかどうかを検査する．起動条件式を評価した結果が真ならば，応



（a） メニュー領域の指定



（b） コマンドポップアップメニューの表示

図 7 A-kit の Text Menu Editor
Fig. 7 Text Menu Editor in A-kit.

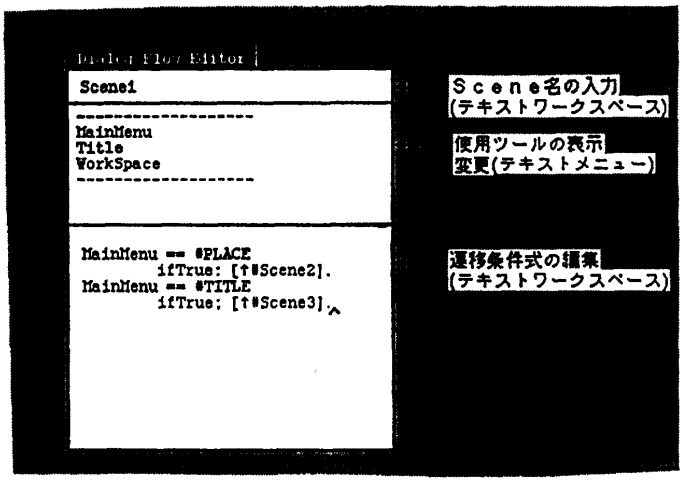


図 8 Scene ネットワークの対話フロー編集
Fig. 8 Editing dialog flow of a Scene network.

用プログラム起動式を評価して応用プログラムを起動する.

5. おわりに

本論文では，ユーザが複数の対話ツールを同時使用できるユーザインタフェース管理システム GUIDMAS の構造と機能，および Smalltalk による実現について述べた．GUIDMAS の設計方針は，UIMS の従来の Seeheim モデルに基づくとともに，Smalltalk の MVC 構造にも対応させることである．また，グラフィックスを積極的に用いる直接操作型のユーザインタフェースの利用を容易にしようとしている．さらに，GUIDMAS の機能を有効に利用できるよう対話の流れの表現 Scene ネットワークと，インタフェース設計者が容易に対話ツールを開発できるユーザインタフェース開発支援システム A-kit を提供している.

UIMS と応用の分離に関連する研究課題のひとつは，意味フィードバック計算をどちら側で行うかである．GUIDMAS は，応用が意味フィードバックを計算する構造である．すなわち，GUIDMAS は，応用との通信のためにコンテナ・オブジェクトを用いる．コンテナ・オブジェクトは共有オブジェクト SharedObject を持ち，現在活性化している対話ツールの入力状態を応用に知らせる．応用は SharedObject 中のモデル（対話ツール）を操作して意味レベルのフィードバック計算を行い，GUIDMAS の対話ツールがアプリケーションインタフェースを通して応用からのメッセージを受け取る.

迅速なシステムのフィードバック応答を実現するためには，本論文で仮定する UIMS と応用間の通信速度を高速にするという考え方とは別に，UIMS 側に意味計算機能を持たせる，またはオブジェクト指向を徹底して Seeheim モデルとは異なる UIMS 構造を導入することが考えられる.

本論文では，UIMS の構造と機構についてひとつの提案をしたが，今後，システムおよびインタラクションのパラメータについて実験して UIMS の有効性を評価することは重要である．GUIDMAS を開発したワークステーション (Tektronix 4404) では，Smalltalk-80 と他言語 (C, Fortran, Lisp など) を接続できるので他言語で作成された応用システムと GUIDMAS との結合は可能である.

これまで提案されている UIMS の対話制御の型には，応用主導，UIMS 主導，および相互独立の 3 つがある[3]．GUIDMAS は，UIMS 主導型に属する制御方法をとる．すなわち，GUIDMAS は，起動条件式と起動式を用いて対話の流れを制御する．アプリケーションプロファイル中にある起動条件式および起動式は，応用の起動だけでなく，入力情報の監視，すなわち入力誤りに対するエラーメッセージ表示と再入力をユーザに促し応用に無効なデータを送らないようにも用いられる．さらに，起動条件式で複数対話ツールに関して記述することにより，複数対話ツールによる応用との対話を可能としている.

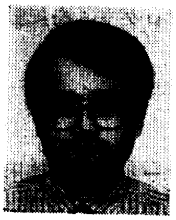
今後の課題としては，アプリケーションインタフェースの仕様設計がある．特に，応用主導型/UIMS 主導型のどちらでも使える相互独立型[3] にするためのアプリケーションインタフェースとアプリケーション間の通信方法が課題である（たとえばセマフォアを用いて，共有オブジェクトの考え方をさらに進めることなど）．また，現在の GUIDMAS と A-kit では，アプリケーションプロファイル，Scene ネットワークの遷移条件式の記述に Smalltalk-80 のメッセージ式を用いているが，これをより簡単に指定できるようにする必要がある．さらに，ユーザインタフェース開発とエンドユーザの利用をより容易にするために，知識ベース（エキスパートシステム）機能を備える知識ベース UIMS は重要な研究課題である.

参考文献

1) Shneiderman, B.: *Designing the User Interface*, Addison-Wesley (1987).
2) Betts, B. et al.: Goal and Objectives for User Interface Software, *Comput. Gr.*, Vol. 21, No. 2, pp. 73-78 (1987).
3) 今宮淳美：ユーザインタフェース管理システム，情報処理ハンドブック，第 13 編，第 10 章，pp. 1194-1201，オーム社 (1989).
4) Lowgren, J.: History, State and Future of User Interface Management System, *SIGCHI Bulletin*, Vol. 20, No. 1, pp. 32-44 (1988).
5) Goldberg, A. and Robson, D.: *SMALLTALK-80: The Language*, Addison-Wesley (1983).
6) Hill, R. D.: Supporting Concurrency, Communication, and Synchronization in Human-Computer Interaction—The Sassafras UIMS, *ACM Trans. Gr.*, Vol. 5, No. 3, pp. 179-210 (1986).
7) Levine, H. and Rheingold, H.: *The Cognitive Connection*, Prentice-Hall (1987).
8) Foley, J. and VanDam, A.: *Fundamentals of*

Interactive Computer Graphics, pp. 55-87, Addison-Wesley (1982).
9) Jacob, R. J. K.: A Specification Language for Direct-Manipulation User Interfaces, *ACM Trans. Gr.*, Vol. 5, No. 4, pp. 283-317 (1986).

(平成元年6月6日受付)
(平成2年2月13日採録)



今宮 淳美 (正会員)

1945年生. 1968年東北大学通信工学科卒業, 1973年東北大学博士課程修了. 工学博士. 1973年東北大学助手, 1975年山梨大学計算機科学科助教授. 現在, 山梨大学電子情報工学科教授. 研究：図形処理と対話システム, ヒューマンインタフェース, アルゴリズム. ACM, IEEE, 人工知能学会, AAAI 各会員.

関村 勉 (正会員)

1962年生. 1987年山梨大学計算機科学科卒業, 1989年山梨大学修士課程修了. 現在, 富士通(株) SE テクニカルセンタ所属. 在学中の研究テーマは, オブジェクト指向 UIMS ソフトウェア.