ヒューマンインタフェース 28－4
（1990．1．18）

ユーザインタフェース構築用イベント駆動型言語の開発

川島正徳　今宮淳美　坂本忠明

山梨大学　電子情報工学科

ユーザインタフェース管理システム（UIMSs）は，ユーザインタフェースの設計と実現を支援する．殆どの UIMSs において，ユーザインタフェース設計者は言語または文法でインタフェースを記述する．これらのシステムでは，入出力動作の形式的手順のような，インタフェースの構文レベルを記述するために言語を使用する．

私達の目的は，インタフェースの並列動作を可能にするために，同時に使用可能な複数の入力装置をエンドユーザへ提供することである．本論文では，マルチプロセス環境においてユーザインタフェースを構築するための UIMS “MUTE-System”，およびマルチスレッド形式でイベントベースの対話を記述するためのイベント駆動型言語 “Mute” について述べる．MUTE-System は Seeheim モデル[7,8,9]を，Mute 言語はＣ言語を基礎としている．

DEVELOPMENT OF EVENT-DRIVEN LANGUAGE FOR CONSTRUCTING A USER-INTERFACE

Masanori KAWASHIMA　Atsumi IMAMIYA　Tadaaki SAKAMOTO

YAMANASHI University, Department of Electrical Enginnering & Computer Science

The User-Interface Management Systems, UIMSs, allow to design and implement user-interfaces. In most UIMSs, the user-interface designer specifies the interface such as the legal sequence of input and output actions with a special-purpose language or grammar.

Our purpose is development of UIMS that provides multi-thread input devices to the end-user, these devices allow s/he to perform concurrent interaction. This paper describes MUTE-System (Multi-Threaded Event-driven System) for developing a user-interface in a multi-process execution environment, and an event-driven language called Mute for specifying multi thread and an event-based dialogue. The MUTE-System is based on the Seeheim model[7,8,9], and the Mute language is based on C.

1 はじめに

ユーザインタフェース (UI:User Interface) 支援環境に関する研究には大きく分けて，人間の側面に関する情報の提供や獲得方法に注目する認知科学からのアプローチと，計算機の OS 上に構築するシステム的な側面からのアプローチの２種類が考えられる[1]．本研究では，後者を扱う．

人間の思考の手助けとして，図式化の有効性が評価されている．しかし，グラフィカル UI は，グラフィカルでない UI に比べ記述が難しい．なぜなら一般に，グラフィカルシステムは様々な入力装置 ── キーボード，マウス，タブレット，ジョイスティック等 ── を複数装備しており，それらの装置の並列使用の支援は欠かすことができないためである．そこで複数の入力装置を支援するグラフィカル UI をより簡単に記述するためのユーザインタフェース管理システム (UIMSs: User Interface Management Systems) に関する様々な研究が行われてきた[1,...,12]．

本研究の目的は，複数入力装置の支援と，複数のアプリケーションの同時実行 (マルチスレッド対話:Multi-thread dialogue) の支援を行う UI の提供である．そこで，エンドユーザ，UI，アプリケーション間のイベント伝達に注目し，イベントの送受信，加工等を行うための言語を開発し，その言語を UI 機能に組み込むための統合的な UIMS を構築する．

2 研究の対象

本研究では，ハードウェア支援とソフトウェア支援により，UI を統合的に管理・支援する．

ハードウェア支援には，複数入出力装置支援がある．具体的には，入力装置支援としてマウス，キーボード，三次元ジョイスティックを支援し，また他の入力装置を支援するための拡張性も持つ．出力装置の支援としては，三次元ウィンドウ処理を対象としているが，マルチスクリーン環境への適応も可能である．

ソフトウェア支援には，UI の動作効率を考え，イベント駆動型のマルチスレッド環境を提供し，基本的な相互作用技法 (interaction technique) をテンプレートとして与える．また，これらを利用するための機能も提供する．

3 ユーザインタフェースに関する研究

UI を支援する機能は，基本的に Seeheim 会議での合意があり，これまでに研究・開発されている UIMSs は，この Seeheim モデル[7,8,9]を基礎としている．

3.1 ユーザインタフェースの Seeheim モデル

UI のモデルとして，Seeheim モデル (図１参照) がある．このモデルはかなり一般的で，既存の UIMSs のいくつかを適切に記述しており，特定の UIMSs に依存しない数少ないモデルの１つである．本研究の目的は，このモデルを複数のユーザが複数のアプリケーションを使用するための UI にまで拡張することである．

Seeheim モデルは，ユーザインタフェースを３つの要素，プレゼンテーション要素 (PC:Presentation Component)，対話制御要素 (DCC:Dialogue Control Component)，アプリケーションイ

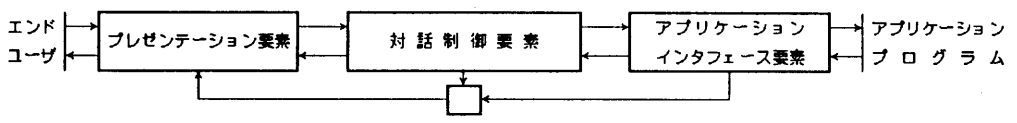

図１ ユーザインタフェースの Seeheim モデル

ンタフェース要素（AIC:Application Interface Component）に分割し，各要素に対するフィードバック機能を持つ．

PC は，UI における物理的表現を扱い，入出力装置，スクリーン配置，相互作用技法，ディスプレイ技法等を含む．PC は，入力装置により直接操作できる唯一の要素であり，他の要素は，直接対話することはできない．PC は，エンドユーザとコンピュータの言語モデル[2]における字句レベル（lexical level）となる．

DCC は，エンドユーザとコンピュータシステム間の対話を扱う．DCC は，言語モデルの構文レベル（syntactic level）となる．

AIC では，UI とアプリケーションプログラム間のインタフェースを記述する．AIC は，言語モデルの意味レベル（semantic level）となる．

これら３要素は論理的に独立したプロセスであり，各要素はトークン(token) により通信する．トークンは，あらかじめ定めた名前と関連するいくつかのデータ値を持つ．

殆どの UIMSs で強調しているのは DCC である．DCC を記述するためのモデルとして，遷移ネットワーク（transition networks），文脈自由文法（context-free grammars），イベント（events）等がある．Green は，このうち記述力が最も優れているのはイベントであり，他の手法によるモデルは全てイベントモデルに変換することができることを示した[8]．

3.2 対話制御要素のイベントモデル

イベントモデルは，いくつかのグラフィックスパッケージにみられる入力イベントの概念に基づいている．これらのパッケージでは，入力装置をイベント源とみなす．各入力装置は１つまたは複数のイベントを，エンドユーザが入力装置と相互作用（interaction）するときに生成する．イベントは，発生原因となった相互作用を特徴づける名前または番号と，そのイベントに関連するいくつかのデータ値を持つ．例えば，タブレットの場合，カーソルが移動した各時間に move イベントが発生する．このイベントに関するデータ値は各時間におけるカーソルの x，y 座標である．

イベントモデルは入力イベントの概念の拡張であり，各要素間および DCC 内における遷移は，イベントの送受信により変遷するという概念に基づいている．イベントモデルは，任意のイベント型（event type）を持つ．UI 設計者は，個々のアプリケーションに対し，より適切なイベント型を自由に定義することができる．発生したイベントは，１つまたは複数のイベントハンドラ（EH: Event Handler）へ送信される．EH は，ある特定のイベントタイプを処理することができるプロセスである．EH は，処理可能なイベントを１つ受信すると，そのイベントに関連するプロシージャ（EHP:Event Handler Procedure）を実行する．EHP では，計算，新しいイベントの生成，アプリケーションの実行，新しい EH の生成，既存の EH の削除等の処理を行なう．

EH の動作は，テンプレートにより定義する．テンプレートは，処理可能なイベント，局所変数，初期設定，そして受信したイベントに関連して実行する処理を記述する EHP を定義するセクションに分割できる．実際の EH のインスタンスは，テンプレート名とその EH を特徴づけるための初期値により生成し，生成した EH には，参照するための一意な名前または番号が与えられる．同一のテンプレートから生成した個々の EH のインスタンスは，各々異なる局所変数（パラメータや変数値）を持つことができる．生成された EH は，任意の EH に削除されるまで活性化状態にある．

イベントモデルでは，UI の DCC を EH を定義するテンプレートの集合により記述する．このモデルでは，システムの実行時に１つのテンプレートから DCC の中心となる EH のインスタンスを生成し，この EH は他の全ての EH のインスタンスを，直接または間接的に生成する．概念的に全ての EH は並列に動作する．

UI 中のイベント発生源が，２種類ある．１つは UI の残りの要素，PC と AIC である．これら

の要素はトークンを DCC へ送信する．もう１つの発生源は DCC 内部である．EH は，自分自身または他の EH へ新しいイベントを生成し，送信できる．これによりEH 間の通信が可能となる．

3.3 マルチスレッド型のユーザインタフェース

マルチスレッド型の UI は，実行可能な個々のアプリケーションに対応する UI 要素をいくつかのスレッドに分割し，各スレッドへデータを送信することで対象とするアプリケーションを実行する．ここで，各スレッドの入口は複数の入力装置である．複数の入力装置を支援する環境では，エンドユーザはどの入力装置を使用するかを指定する必要はない．完全な並列入力を実現するためには，単一アプリケーションへの複数の入力装置からの入力に対する支援が必要である．そこで，各スレッドは，複数の入力装置からのデータを受信し，また，他のスレッドに対して字句的（lexically）に，そして構文的（syntactically）に異なる方法で処理されなければならない．また，他のスレッドに対して論理的に非同期であるためには，マルチタスキング（multi-tasking）の支援が必要となる[11,12]．

4 MUTE-System の開発

本研究では，これまでの研究に基づく，複数の入出力装置の使用と，複数のアプリケーションの実行を支援する UI を開発する．

開発するイベント駆動型の UI を MUTE-System (MUlti-Threaded Event-driven System) と呼び，基礎となる UIMS のモデルを MUTE-Model と呼ぶ．

4.1 MUTE-Model の構造

複数の入力装置，複数のアプリケーションを支援するための UIMS モデル，MUTE-Model，を図２に示す．このモデルは Seeheim モデルにうまく適合する．各構成要素を，Seeheim モデルと対応づけると，以下のように分類できる．

●PC:
- 入力要素（IC:Input Component）
- 出力要素（OC:Output Compoment）

●DCC:
- スイッチボード（SWB:SWitchBoard）
- ダイアログイベントハンドラ（DEHs:Dialogue Event Handlers）

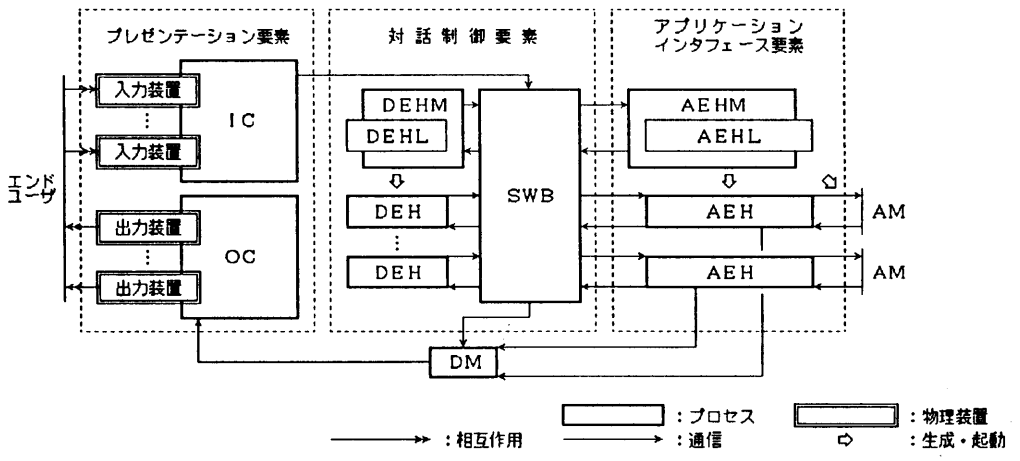

図２ MUTE-Model の構造

・ダイアログイベントハンドラマネージャ (DEHM:Dialogue Event Handler Manager)

●AIC:

・アプリケーションイベントハンドラ (AEHs:Application Event Handlers)

・アプリケーションイベントハンドラマネージャ (AEHM:Application Event Handler Manager)

●フィードバック:

・ディスプレイマネージャ (DM:Display Manager)

以下で，Seeheim モデルに対応する要素ごとに，各構成要素の機能について説明する．

4.1.1 MUTE-System の PC

●IC:

エンドユーザと入力装置間の相互作用を検出するために，使用可能な各入力装置の状態をビッグループ手法 (Big Loop method)[12] により監視する．そして1サイクル前の状態と比較し，状態変化が検出されたなら相互作用があったものとみなし，その相互作用を意味するトークンと関連するデータ値を DCC へ送信する．

●OC:

DM から文字や図形の情報を受信し，複数の出力装置への分配，および各出力装置の手続きに従った画面表示を行う．

4.1.2 MUTE-Model の DCC

●SWB:

DCC 内および各要素間のデータ (トークン，イベント，データ値) の流れを制御する．他の要素から SWB へのデータは，全てキュー (queue) に蓄えられる．SWB はキューからデータを順次読み込み，そのデータに付随する情報と，DEHM と AEHM が共有するイベントハンドラ状態テーブル (EHST:Event Handler Status Table) の情報により送信先となる DEHs や AEHs を決定し，データを送信する．EHST は，現存する DEHs と AEHs の番号，対応するウィンドウ番号，各 EH の状態等の情報を持つ．

●DEHs:

イベントまたはトークンを受信し，それらに関連する処理を行う．トークンを受信した場合は，DEHs のテンプレートで宣言したイベントに変換し，DEHs 内での処理は全てイベント単位で行う．

●DEHM:

DEHs のインスタンスを生成する．また，DEHs の番号や状態，ウィンドウとの対応等を EHST へ登録する機能を持つ．

DEHs は，DEHM が持つダイアログイベントハンドラライブラリ (DEHL:Dialogue Event Handler Library) 内の DEHs のテンプレートから生成される．DEHs のテンプレートは，システム DEHs (SDEHs:System DEHs) と，ユーザ DEHs (UDEHs: User DEHs) に分類することができる．SDEHs は，UI 設計者へ提供する基本的な UI 要素の機能を持つ EH であり，UI 設計者は，必要に応じてこれらの EH を自由に参照することで，目的とする EH を構築することができる．例えば，SDEHs の1つであるスライダ (slider) SDEH の機能は，次のイベント名と属性を与えることで利用できる．

イベント名：SLIDER1
属性　　：表示位置 (xHome, yHome)
　　大きさ (xSize, ySize)
　　メモリの範囲 (下限，上限)
　　初期値 (メモリ範囲内の値)
　　メモリの刻み (何等分か)
　　向き (0:縦表示，1:横表示)

UDEHs は，UI 設計者が作成し，登録した DEH である．UI 設計者が UDEHs を記述するために，イベント駆動型の Mute 言語を提供する．Mute 言語によるスライダ SDEH の記述例を図3に示す．

```
dialogue event handler SLIDER1 is
  token
    MMOVE   sMove;
    MB1ON   on;
    MB1OFF  off;
  end;
  constant
    integer VTYPE = 0;
  end;
  variable
    integer sType, zero, nowV, dS1,
            gSize, state;
  end;
  initial
    argument
      integer posX, posY, sizX, sizY,
              minV, maxV, iniV, gage, type;
    end;
    process
      nowV  := iniV;   state := 0;
      zero  := posX;   sType := type;
      if type = VTYPE do
        gSize := sizY/gage; zero := posY;
      else do
        gSize := sizX/gage; zero := posX;
      end;
      dS1 := call "drawSlider1" with
        posX, posY, sizX, sizY,
        minV, maxV, iniV, gage, type;
    end;
  end;
  event on do
    process state := 1; end;
  end;
  event off do
    process state := 0; end;
  end;
  event sMove do
    argument integer x, y; end;
    variable integer newV; end;
    process
      if state = 1 do
        if sType = VTYPE do newV := y;
        else             do newV := x; end;
        newV := (newV-zero)/gSize;
        if nowV <> newV do
          nowV := newV;
          send event "redraw"
            with nowV to dS1;
          send event "slider1"
            with nowV, self to parent;
        end;
      end;
    end;
  end;
  final
    process kill dS1; end;
  end;
end SLIDER1.
```

図3　スライダ SDEH の記述例

UI 設計者は，次に示す文を記述することで図4に示すスライダ SDEH を生成することができる．

```
create "SLIDER1" with xHome, yHome,
                      xSize, ySize,
                      0, 10, 5, 11, 1;
```

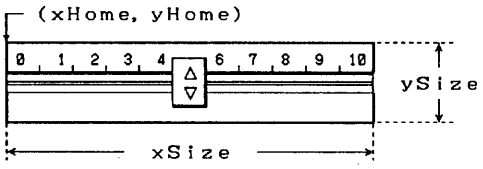

図4　スライダ SDEH の使用例

4.1.3 MUTE-Model の AIC

●AEHs:

UI とアプリケーションモジュール (AM: Application Module) 間の通信プロトコルを定義し，それに従ったデータの変換を行う．

●AEHM:

AEHs のインスタンスを生成する．また，DEHM 同様に，EHST への登録を行う．

AEHs も DEHs と同様に，AEHM が持つアプリケーションイベントハンドラライブラリ (AEHL: Application Event Handler Library) 内のテンプレートから生成される．

4.1.4 MUTE-Model のフィードバック

●DM:

表示に関する前処理を行う．SWB と AEHs からの表示要求を受け，表示対象となるウィンドウに合わせて文字の折り返しや，図形のクリッピング処理を行う．処理後のデータは，OC へ送信する．

また，DEHs がウィンドウを持つ場合は，DEHs の生成・削除時に，ウィンドウのオープン・クローズ処理を行う．

4.2 MUTE-System の構築

MUTE-System の構築環境は，UNIX 4.3 BSD 上で MakeMUTE コマンドを実行することで設定でき，以後恒久的に使用することができる．一方，各 AM に対応する UI 要素を記述するために，UI 設計者は Mute 言語を使用し，UI 記述を構築環境内のコンパイラを通すことで MUTE-System に組み込む．この時点でエンドユーザは UI 要素を加味したアプリケーションを使用することができる．

5 Mute 言語の開発

UI 設計者が DEHs と AEHs を記述するための言語として，イベント駆動型の Mute 言語を提供する．以下では，DEHs 用 Mute 言語と AEHs 用 Mute 言語，そして Mute 言語コンパイラについて述べる．

5.1 DEHs 記述用の Mute 言語

DEHs を記述するための Mute 言語の構造と使用できる文について述べる．DEHs を記述したファイルには，拡張子 ".deh" を付けることで他のファイルと区別する．

5.1.1 DEHs 用 Mute 言語の構造

DEHs 用 Mute 言語は，6宣言部から構成される．

●ウィンドウ宣言部：

DEH がウィンドウを持つ場合に，そのウィンドウの属性を宣言する．DEH がウィンドウを持つかどうかは，ウィンドウ宣言部の有る無しで区別する．ウィンドウの属性として，以下の項目が挙げられる．ただし，特に指定しない場合は既定値が与えられる．

- タイトル
- タイトル色
- 大きさ (xSize, ySize)
- 境界線色

●トークン宣言部：

DEH が使用可能なトークンを宣言し，DEH 内の処理単位であるイベントに対応付ける．複数のトークンを同一のイベントに対応付けることもできる．

●定数宣言部：

DEH 内で使用する定数を宣言する．この定数は，宣言をした DEH のテンプレート内のみで有効となる．

●変数宣言部：

DEH 内で使用する変数（＝レジスタ）を宣言する．この変数も定数と同様に宣言をした DEH のテンプレート内でのみ有効である．

●初期処理宣言部：

DEH が生成されたときに最初に実行する処理を宣言する．初期処理宣言部は，更に以下の宣言部に分割できる．

- 引数宣言部：

  DEH のインスタンス生成時に，DEH を特徴付けるために与えられる初期値を引数として宣言する．引数はある名前と型を持ち，各々 create 文（後述）の引数の並びに順に対応する．

- 変数宣言部：

  初期処理宣言部内でのみ有効な局所変数を宣言する．ここでの宣言は，DEH 内で有効な変数の宣言に優先する．

- 処理宣言部：

  初期処理宣言部で行う処理を Mute 言語の文（後述）により記述する．

●イベント宣言部：

イベント宣言部は宣言部名を持ち，DEH が受信したイベントを受信すると，そのイベント名と同一の名前を持つイベント宣言部が実行される．イベント宣言部では，受信した個々のイベントに関連して実行する処理を宣言する．この宣言部は，DEH が処理可能なイベントの数だけ存在する．イベント宣言部は，更に以下の宣言部に分割できる．

- 引数宣言部：

イベントに関連して送られるデータ値を引数として宣言する．これは send 文の引数の並びに順に対応する．

・変数宣言部：

イベント宣言部内でのみ有効な局所変数を宣言する．ここでの宣言は，DEH 内で有効な変数の宣言に優先する．

・処理宣言部：

受信したイベントに関連して実行する処理を Mute 言語の文で記述する．

●終了処理宣言部：

終了処理宣言部は，DEH が削除されるときに実行する処理を記述する．終了処理宣言部は，更に以下の宣言部に分割できる．

・変数宣言部：

終了処理宣言部内でのみ有効な局所変数を宣言する．ここでの宣言は，DEH 内で有効な変数の宣言に優先する．

・処理宣言部：

終了処理宣言部で行う処理を Mute 言語の文により記述する．一般に，DEH 内の他の宣言部で生成・起動した子 DEH の削除や AM の強制終了を行う．

5.1.2 DEHs 用 Mute 言語の文

DEHs を記述するために，以下の文を使用することができる．

●代入文 (assignment 文)：

新しい値を計算し，レジスタに格納する．

●create 文：

引数として与えられる DEH のテンプレートと，DEH を特徴付けるための初期値から，新しい DEH を生成する．生成が成功すると，新しい DEH のインスタンス番号を返す．

●destroy 文：

与えられたインスタンス番号により，既存の DEH を削除する．

●send 文：

新しいイベントまたはトークンを生成し，関連するデータ値と共に他の要素へ送信する．

●call 文：

与えられた AM 名により，AM を起動する．AM の起動と，対応する AEH の生成が成功すると，AEH のインスタンス番号を返す．

●kill 文：

与えられたインスタンス番号により，対応する AEH の削除，AM の強制終了を行う．

●hold 文：

指定した入力装置からの入力ストリームを確保する．この文の実行後は，SWB のデータ振り分け条件によらず，指定した入力装置からのデータを受信する．

●release 文：

入力装置からの入力ストリームを解放する．

●wait 文：

指定したイベントを受信するまで待つ．

●if 文，case 文：

条件による文の選択実行を行う．

●repeat 文，for 文：

文を繰り返し実行する．但し，有限時間内に処理が終了するための条件が加えられる．

●escape 文：

case 文，repeat 文，for 文中から，処理を中断し，抜け出す．

●skip 文，label 文：

文の飛び越し実行を行う．skip 文を実行すると，skip 文のラベル名と同一のラベル名を持つ label 文が現れるまで文を飛び越す．

5.2 AEHs 記述用の Mute 言語

AEHs 記述用 Mute 言語の構造と使用できる文について，DEHs 用 Mute 言語と比較して述べる．AEHs を記述したファイルには，拡張子 ".aeh" を付けることで他のファイルと区別する．

5.2.1 AEHs 用 Mute 言語の構造

AEHs 用 Mute 言語は，DEHs 用の Mute 言語とほぼ同じであるが，以下の点が異なる．

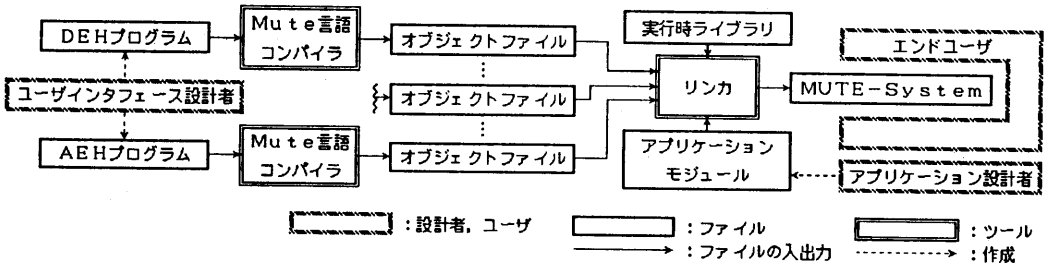

図5　MUTE-System の構築手順

●ウィンドウ宣言部を持たない：
AEH はウィンドウを持たないため，ウィンドウ宣言部はない．

●アプリケーション宣言部を持つ：
AEH が対応することができる AM の名前を宣言するためのアプリケーション宣言部を持つ．1つの AEHs が複数の AM に対応することもできる．

●初期処理宣言部の引数宣言部の相違：
AEH の初期処理宣言部で宣言する引数は，DEHs 用 Mute 言語の call 文の引数に順に対応する．

●終了処理宣言部を持たない：
AEH は，子 EH の生成や AM の起動を行わないため，DEHs でそれらの削除・終了のために設けた終了処理宣言部が不用となる．

5.2.2 AEHs 用 Mute 言語の文

AEHs を記述するために，以下の文を使用することができる．

●代入文 (assignment 文)：
●if 文，case 文：
●repeat 文，for 文：
●escape 文：
●skip 文，label 文：
●send 文：
●wait 文：
以上，DEHs 用 Mute 言語に同じ．
●data 文：
データを対応する AM へ送信する．
●display 文：
DM へ文字や図形の表示要求を行う．

5.3 Mute 言語コンパイラの開発

Mute 言語で記述したプログラムは，Mute 言語コンパイラによりコンパイルする．Mute 言語コンパイラは，入力ファイルの拡張子により対応するコンパイラ，DEH コンパイラ，AEH コンパイラ，を起動する．コンパイルが成功すると，DEH，AEH をそれぞれ DEHL，AEHL へ登録する．

図5に UI 設計者が作成した EH のコンパイルから，実際に UI 要素により AM を使用するまでの過程を示す．リンカは，DEHL，AEHL を参照し，必要なオブジェクトファイルを結合し，アプリケーションの実行環境を構築する．MUTE-System は，このような一連の作業を支援し，動作確認や修正用のデバッグ機能をもつ UIMS となっている．

6 MUTE-System の特徴

MUTE-System の特徴を示す．

●複数の装置を同時に使用することができる．
●DCC を DEHs の集合として定義する．
●AIC を個々の AM に適合する AEH の集合として定義する．
●DEHs，AEHs を記述するための，イベント駆動型の言語 Mute を提供する．

●エンドユーザからの入力を，対応する個々の DEH により非同期に処理する（マルチスレッド）.
●複数の AM を同時に実行することができる.
●AM 間の関係および通信を DEH により管理することができる.
●AM と UI 間の通信プロトコルを AEHs で定義することで，任意の AM に対応することができる.
●Seeheim 会議で取り上げられた事柄を，全て取り入れている.
●様々な相互作用技法が DEHs のテンプレートとして存在し，これを利用することで容易に UI を構築・使用することができる.

7 おわりに

本研究は，現在までに研究・開発されてきた種々の UIMSs を統合して実現することを到達目標としている．そのため，これまで様々な観点から提案されている UIMSs の重要な要素を抽出し，ひとつのシステムとして MUTE-System を構築した．また，AM に対応する UI 要素を記述するための Mute 言語とその支援環境を設計・作成した.

設計・作成過程において，様々な問題が生じた．例えば，開発環境である UNIX 4.3 BSD 上の問題，グラフィクスシステム（LEX90）に関連する入出力等の問題である．これらの問題については，別の機会に述べる.

参考文献

[1] Shneiderman B.著，東基衛，井関治監訳:
“ユーザー・インタフェースの設計”,
日経 BP 社，1987.

[2] 今宮淳美:
“ユーザインタフェース管理システム”,
オーム社，May 1989，情報処理ハンドブック第3編，第 10 章.

[3] Anson, E.:
“THE DEVICE MODEL OF INTERACTION”,
ACM Computer Graphics, Vol.16, No.3, July 1982, 107-114.

[4] Cardelli, L., Pike, R.:
“Squeak:a Language for Communicating with Mice”, ACM Computer Graphics, Vol.19, No.3, July 1985, 205-213.

[5] Coutaz, J.:
“Abstractions for User Interface Design”,
IEEE Computer, Vol.18, September 1985, 21-34.

[6] Flecchia, M.A., Bergeron, R.D.:
“Specifying Complex Dialogs in ALGAE”,
ACM CHI+GI 1987, 229-234.

[7] Green, M.:
“The University of Alberta User Interface Management System”, ACM Computer Graphics, Vol.19, No.3, July 1985, 205-213.

[8] Green, M.:
“A Survey of Three Dialogue Model”,
ACM Transaction on Graphics, Vol.5, No.3, July 1986, 244-275.

[9] Hartson, H.R., Hix, D.:
“Human-Computer Development : Concepts and System for Its Management,” ACM Computing Surveys, Vol.21, No.1, March 1989, 5-92.

[10] Hill, R.D.:
“Event-Response Systems - A Technique for Specifying Multi-Threaded Dialogues”,
ACM CHI+GI 1987, 241-248.

[11] Tanner, P.P, MacKay, S.A., et al.:
“A Multitasking Switchboard Approach to User Interface Management”, ACM Computer Graphics, Vol.20, No.4, August 1986, 241-248.

[12] Tanner, P.P.:
“Multi-Thread Input”, ACM Computer Graphics, Vol.21, No.2, April 1987, 142-145.