ユーザインタフェース作成支援システムの設計と試作について†

竹村治雄†† 辻野嘉宏††
荒木俊郎†† 都倉信樹††

最近，計算機のユーザインタフェース（UIF）の向上の必要性に対する認識が深まりつつある．本論文ではアプリケーションプログラム（AP）における良好なUIF提供の妨げとなる種々の原因をあげ，それらの問題を解決し，APでの良好なUIF提供を可能にするユーザインタフェース作成支援システムUISEを提案し，UISEの設計方針，機能，実現手法について報告している．UISEは，従来APごとに設計されていたUIFを一括して管理し，この一括管理されたUIFをUIF記述言語UIDLを用いてAPから利用することで，比較的高度なUIFを，従来の手法による場合より少ない手間で実現できることを目ざしたものである．また，同時に複数AP間でのUIFの自然な統一も可能にする．UISEを用いることにより，APのプログラマはAPのUIFを容易に記述でき，APのユーザは統一的で良好なUIFを利用できる．

1. まえがき

近年，計算機の急速な発達と普及により，様々な分野での計算機の利用が推進され，これらの分野で利用可能な多種多様のアプリケーションプログラム（AP）が開発されている．これに伴い，計算機の利用者層も，従来のごく限られた専門家層から広く一般者層へと移り変わり，ユーザインタフェース（UIF）向上の必要性に対する認識が深まりつつある[1),2)]．良好なUIFは，APの利用者（ユーザ）の作業能率を高めユーザの能力を最大限に発揮させると同時に，不必要な操作や注意からユーザを解放することにより，ユーザに対して満足感を与える．しかし，現在のAP開発環境では，良好なUIFが提供されるか否かは，APの作成者自身（プログラマ）に委ねられており，プログラマがUIFの向上に無関心であったり，向上のための適切な手法を見いだせなかった場合，UIFの向上は望めない．また，UIFの向上を図ると通常プログラミングコストが急激に増加する．

本論文では，APにおける良好なUIF提供の妨げとなる原因をあげ，それを解決し良好なUIF提供を目ざすユーザインタフェース作成支援システムUISE (User Interface Support Environment)を提案する[3)]．以下，第2章で，良好なUIF提供の妨げとなる原因について考察し，第3章で，それに基づき試作を行ったUISEについて，第4章で，UISEで用いるUIF記述言語UIDLとこれを用いたUIF記述について，第5章でUISEの実現についてそれぞれ述べる．

2. 良好なUIF提供の問題点と解決の一手法

APにおけるUIFの改良を妨げる原因としては，

① 計算機の能力的な制限，
② プログラマのUIFの重要性に関する認識不足，
③ プログラマのUIF提供手法に関する知識の欠如，
④ AP開発コストの制限によるUIF部分の切り捨て，
⑤ ユーザのUIFに関する要求がプログラマに伝わらないこと，

などが考えられる．しかし，①，②については，ハードウェア技術の進歩や，UIFの重要性に関する認識の深まりにより解決されつつある．しかし，③～⑤については，適当な解決法を考える必要がある．

また，一つの計算機システム上で複数のAPが利用される場合，これらAP間でのUIFの統一がとれていないと，利用者がしばしば混乱し，計算機システム全体のUIFが悪化する．したがって，少なくとも，同一利用者に使われるAPは，それらの間でのUIFの統一が行われるべきであるが，現在のAP開発環境では，開発がAPごとに独立して行われているため，UIFの統一は非常に困難である．

以上を総合し，良好なUIFの提供には，

① プログラミングの量はできるだけ増やしたくない，
② 自然にAP間でのUIFを統一したい，

† UISE—User Interface Support Environment by HARUO TAKEMURA, YOSHIHIRO TSUJINO, TOSHIRO ARAKI and NOBUKI TOKURA (Department of Information and Computer Sciences, Faculty of Engineering Science, Osaka University).

†† 大阪大学基礎工学部情報工学科

③ ユーザがある程度 UIF を管理し UIF の不備を補ったり，個人の好みに対応できるようにしたい，

などの要求が生じる．UISE は，以上を考慮して，

① 良好な UIF の実現に必要な諸機能を，ユーザインタフェース管理システム UIMS にまとめ，

② AP とユーザの間に UIMS を置くことにより AP 間の UIF の統一をはかり（図 1），

③ UIMS の持つ機能を AP 側から UIF 記述言語 UIDL を用いてできるだけ少ない記述で利用できる

ように考えられたユーザインタフェース作成支援システムである．この考え方では，UIMS をユーザインタフェースを統轄する一種の OS（オペレーティングシステム）と見なしており，UIF に関するすべての資源を管理する．

通常，このような環境を実現する場合，AP側，ユーザ側のどちらに UIF の制御の主導権を与えるかで実現方法が種々考えられるが，本システムでは，プログラマにとって処理のしやすいデータの入出力をセッションと呼ぶ単位に UIDL を用いて定義し，UIMS がデータの入出力をセッション単位で一括して行うこととした．セッション単位の入出力により，一つのセッション内では UIF はユーザの指示に従う UIMS によって管理され，UIDL 記述による AP 側からのコントロールとユーザからのコントロールとを両立させることができ，かつ AP はセッションに関する記述を除き従来どおりのプログラミングが可能である．この実現手法を用いることにより，従来から提案されている，「入出力サブルーチンの統一」や「入出力ルーチン生成言語を用いる手法」などよりユーザからの制御が自由にでき，より柔軟なシステムを構築できる．

UIDL を用いたプログラミングの特徴には，

① プログラムの UIF 部の記述が宣言的に行えるので記述が容易である，

② 比較的高度な UIF を従来の手法で記述する場合より少ない手間で作成できる，

③ プログラマが意識せずに AP 間で UIF を統一できシステム全体の UIF の向上につながる，

④ AP 側での UIF 処理のオーバヘッドが低減される，

⑤ UIF の個別設計や変更が容易である，

などがある．

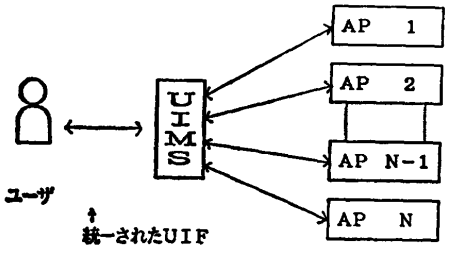


図 1 UIMS を用いたシステムでの UIF
Fig. 1 User interface with UIMS.

3. システムの構成

UISE は，大きくユーザインタフェース管理システム UIMS と，UIDL 処理系（UIDL コンパイラ，テンプレートエディタ等）の二つに分けることができる．システムの全体図を図 2 に示す．本章では，UISE の構成と機能について概略を述べる．

3.1 ユーザインタフェース管理システム UIMS

UIMS は AP とユーザの間に介在し，システムの UIF 全般の管理を行う．UIMS の機能としては，

①入出力データの一括した入出力，②ヘルプ機能，
③自動エラー報告機能，④自動メニュー生成機能，
⑤入出力データ変換機能，⑥デフォルト値設定機能，
⑦日本語入力，⑧マルチウィンドウ機能，
⑨同義語定義機能，⑩マクロコマンド機能

等がある．このうち，②〜⑥の機能は，UIDL による記述をもとに実現される機能である．

UIMS の残りの機能（⑦〜⑩）は，UIDL で陽に記述されていない UIF に関する機能を実現するためのものである．これらは，特に新しい概念ではないが，UIF 向上のために有効であると考えられ，かつ AP ごとの対応による一貫性の欠如をなくすため UIMS の機能として採用したものである．以下では①〜④の機能について簡単に紹介する．

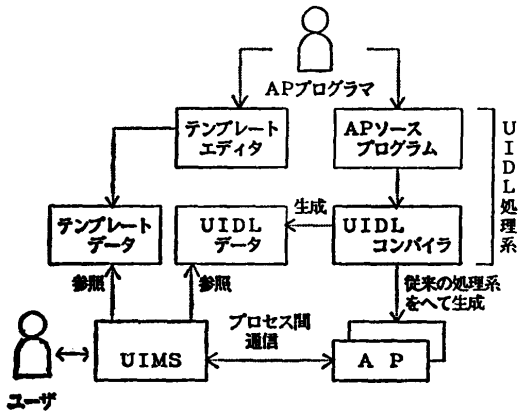


図 2 システムの全体図
Fig. 2 Architecture of UISE.

① 入出力データの一括した入出力

UIDL で記述されたセッション単位の入出力を行う機能である．ユーザは，セッション内ではデータの入力を任意の順序で行うことができる．また AP 側にとっても，内部処理に都合の良いデータ構造に直接データが入出力されるので，入出力のためのデータ構造の用意が不要であり，**図 3** に示されるような試験の成績表を構成する 3 次元の配列の入出力も可能である．ユーザに対しては，3 次元配列のままの形でデータを表示することはできないので，UIMS ではこれをあらかじめ用意されたテンプレートと呼ぶ 2 次元の書式を用いて表示する．したがって，テンプレートを用意すれば，この成績表は，人数×科目の表の試験回数分の入力によっても，試験回数×科目数の表の人数分の入力によっても，試験回数×人数の表の科目数分の入力によっても作成可能である．試験回数×科目のテンプレートを用いて入力を行う場合，人の順番が 3 次元データの一種の切り口となっており，UIDL で宣言されたテンプレート中の切り口を変えることで表示される面を変えて順次入力を行う（**図 4**）．

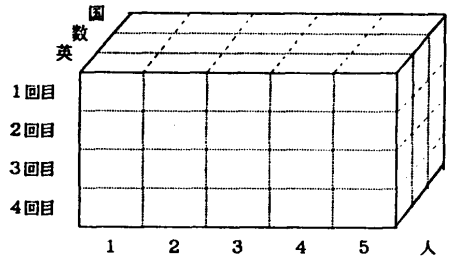


図 3 3 次元の配列データの例
Fig. 3 Example of 3-dimensional array data.

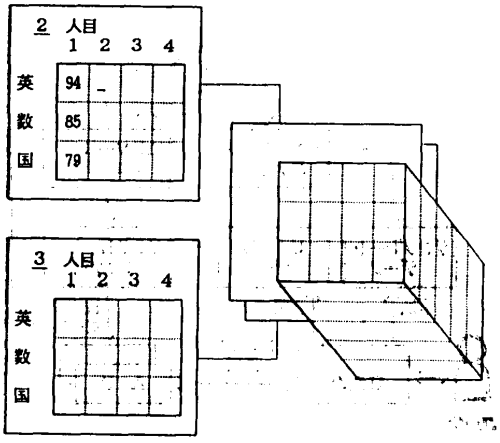


図 4 試験回数×科目での入力
Fig. 4 Input of 3-dimensional array data.

② ヘルプ機能

ユーザがヘルプキーを押すことにより，起動される機能である．UIMS 自身の操作法などのメッセージまたは，AP によって用意されたデータの入力法，コマンドの説明などのメッセージが表示される．UIMS 自身の操作法に関するメッセージは，グローバルヘルプキーが押された場合に，AP によって用意されたものは，ローカルヘルプキーが押された場合に，それぞれ表示される．AP によって用意されるメッセージは，あらかじめ UIDL のメッセージ定義文で定義されたもので，セッション中の現在カーソルが位置しているフィールドで入出力されるデータに割り当てられたメッセージすべてが階層的に表示される．ヘルプメッセージ表示の例を**図 5** に示す．

③ 自動エラー報告機能

ユーザが入力したデータが，AP が要求しているデータの型，範囲と異なる場合に，ユーザに対してエラーの種類，入力すべきデータの型や範囲を表示する機能である．報告されるエラーの種類としては，(1) 入力されたデータの型 AP で要求される型と異なる場合（タイプエラー），(2) 入力されたデータの範囲が AP で要求されるデータの範囲と異なる場合（レンジエラー），(3) 入力されたデータ（複数のデータを含む）があらかじめ設定された条件を満足しない場合（条件エラー），の三通りがある．それぞれのエラーに対するメッセージは，UIDL のメッセージ定義文で定められる．また，データに要求されるタイプとレンジはテンプレートのフィールドに指定されたものであり，複数データ間の条件は，UIDL の条件定義文で定

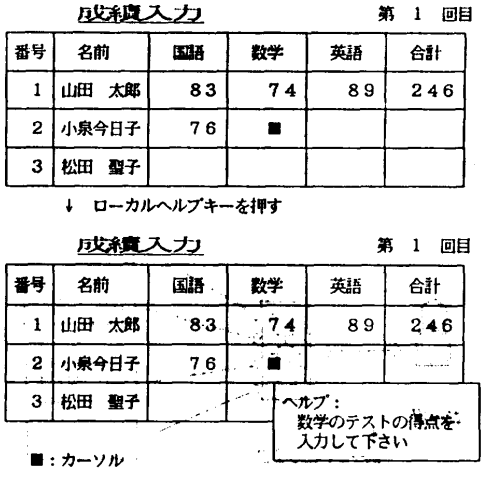


図 5 ヘルプ機能の例
Fig. 5 Example of help function.

められるものである．条件エラーは，条件判定に必要なデータがすべて入力された時点で判断され，報告される．タイプエラー表示の例を**図6**に示す．

④ 自動メニュー生成機能

ユーザがメニューキーを押すことによって，その時点で，ユーザが選択可能な入力をメニューとして表示する機能である．ヘルプキーの場合と同様に，グローバルメニューキーとローカルメニューキーとが存在しそれぞれ，UIMS の操作に関するメニューと AP でのメニューを表示するために使用する．ローカルメニューは，UIDL で記述された要求される入力データが，限られた数以下の入力要素から構成される場合に，ユーザがローカルメニューキーを押すことにより自動的に生成され表示されるものである．ユーザは，表示されるメニューから選択を行うことにより，メニューを用いないで直接入力を行うのと同じに入力ができる．メニュー生成機能の例を**図7**に示す．この例では処理内容として許される入力が TYPE, SORT, DELETE のいずれかであることが，UIDL によって記述されており，ユーザがローカルメニューキーを押すことにより，図のようなメニューが表示され，ユーザはメニューから適当な項目を選択する．

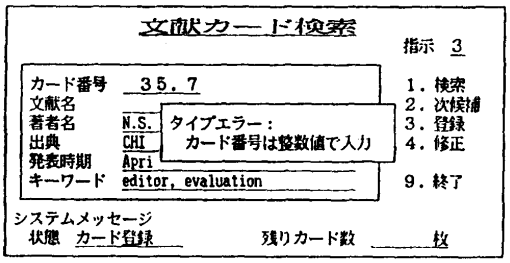


図6 タイプエラーの例
Fig. 6 Example of data type error.

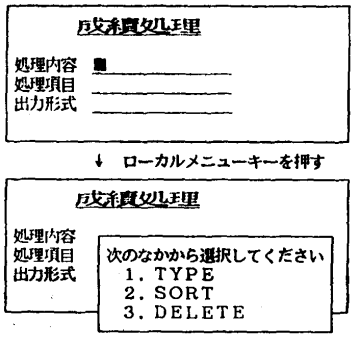


図7 メニュー生成機能の例
Fig. 7 Example of menu function.

3.2 UIDL 処理系

UIDL 処理系は，UIDL による記述を処理する UIDL コンパイラとセッションでの二次元書式記述を作成するための専用エディタ（テンプレートエディタ）で構成される．UIDL による記述は，従来のプログラミング言語（ホスト言語）による AP の UIF 部分以外の記述に埋め込んで用いる．したがって，UIDL コンパイラは，UIDL による記述を含むソースファイルを入力とし，これを処理し，UIDL 記述を含まない従来のコンパイラで処理可能なソースプログラムと，実際に UIDL による記述を実現するために AP 実行時に，UIMS 内の UIDL データインタプリタによって解釈実行される一種の中間コードである UIDL データとを出力する．今回はC言語をホスト言語として用いた．UIDL コンパイラの出力を中間コードとすることにより UIMS の移植性が確保されている．

テンプレートエディタは，UIDL のテンプレート宣言文によって指定されるテンプレートを実際に作成する際に用いるエディタで，テンプレート自身が二次元の書式を記述するという性質から，実際に CRT の画面を見ながらテンプレートがユーザ表示される状況を常に把握しながら作業ができるように画面エディタとして実現されている．テンプレート内のフィールドが持つ各種の属性はモード切換えによって表示されるフィールド設定用画面で設定できる．テンプレートエディタによるテンプレート編集の例を**図8**に示す．図の例では，テンプレートを作成するプログラマは，まず画面上でタイトル，罫線などのテンプレートの固定部分を入力し，さらに可変部分であるフィールドの位置と番号を入力して行く．同じ番号のフィールドが用いられた場合，これらは一定の規則で順序付けられ区別される．さらにカーソルがフィールドを指している状態で「書式キー」を押すことにより，そのフィールドに関しての書式を入力できる．この状態でプログラマは，フィールドの大きさ，入力データの変換ルール，表現形式，許される範囲を入力できる．テンプレートエディタによって作成されたデータは，テンプレートファイルとして保存され，UIDL データファイルとともに入出力の実行時に UIMS によって利用される．

4. ユーザインタフェース記述言語 UIDL

本章では，UIDL と UIDL による記述の具体的な例と特徴について述べる（UIDL の構文図は付録に示す）．

今回設計した UIDL は，既に述べたように C 言語に埋め込んで用いるが，一般に AP の UIF を記述する言語の実現方法としては，ホスト言語とは独立に作る方法とホスト言語に依存して作る方法の二つが考えられる．前者は，入出力ルーチン生成言語の実現等で用いられる手法であるが，前者を用いた場合，UIDL 処理系が容易に作成でき，複数の言語で同じ UIDL を用いることができるなどの利点を持つが，UIDL による記述と従来の言語による記述の間に違和感が生じ，UIDL 記述部分とそれ以外の部分との橋渡し的な記述が必要であるなどの欠点を持つ．一方，後者を用いた場合は，記述の違和感が減り記述量も減らせる反面，処理系を言語間で共通にできず，UIDL も言語ごとに異なったものになる．しかし，通常 AP のプログラマが複数の言語を混在して用いることは少なく，複数言語間の UIDL の統一よりも，単一言語での UIF 記述を減少するほうが，プログラマの負担を減少できると考えられる．また，処理系作成のコストは一時的であるので，今回設計した UIDL は後者のホスト言語に依存したものとした．

UIDL による UIF の記述は，大きく

① 入出力されるデータの実体を定義する部分，
② セッションを定義する部分，
③ 実際の入出力を記述する部分，

に分かれている．以下では，具体例をあげてこれらについて述べる．**図 9** は，40 人の英語，数学，国語のテスト各 5 回分の成績を入力する場合の UIF の記述例である．

まず，①は，UIMS がセッション単位でのデータ入出力を行う場合に，実際に入出力されるデータの実体を定義する部分であり，プログラマが内部処理に都合の良い構造で宣言することにより，そのデータの実体をそのまま入出力用の領域として用いることができ，記述量を減少できる．図では成績表は 3 次元の配列として宣言されている．

次に，②は，UIDL による UIF 記述の中心をなすもので，この定義部には，

（1）セッション名とそのセッションで入出力されるデータを表す仮の名前と構造，
（2）テンプレートの指定とテンプレート内のフィールドと入出力データの割り当て，
（3）入出力データのデフォルト値，

成績入力　　第 ＄6回目

番号	名前	国語	数学	英語	合計
1	＄1	＄2	＄3	＄4	＄5
2	＄1	＄2	＄3	＄4	＄5
3	＄1	＄2	＄3	＄4	＄5

(a) フィールドの指定

成績入力　　第 ＄6回目

番号	名前	国語	数学	英語	合計
1	＄1	＄2	＄3	＄4	＄5
2	＄1	＄			
3	＄1	＄			

フィールド　＄2
フォーマット　###
変換　10進 整数
入出力ルール　右づめ ゼロサプレス
範囲指定　0 .. 100

(b) 書式の指定

図 8 テンプレート編集
Fig. 8 Template editor.

```
/* ①入出力されるデータの実体を定義（C言語による）*/
char namelist[40][12];  /* 名前表 */
int  seiseki[5][3][40];  /* 成績表 */
int  goukei[5][40];     /* 合計の表 */

/* ②セッションを定義  (SIO という名前のセッション記述) */
session SIO(namelist,seiseki,goukei)/* (1)セッション名と仮入出力引数の宣言*/
    char namelist[40][20];
    int  seiseki[5][3][40];      /* 仮入出力引数の構造を定義 */
    int  goukei[5][40];
{
  template tempa (i=0..4) {      /* (2)tempa という名前のテンプレートを指定 */
    $1[0..39] = namelist[0..39],output; /* テンプレートとの対応を定義する */
    $2[0..39] = seiseki[i][0][0..39],input;   /* iは切り口変数 */
    $3[0..39] = seiseki[i][1][0..39],input;
    $4[0..39] = seiseki[i][2][0..39],input;
    $5[0..39] = goukei[i][0..39],output;
    $6 = i,input;
  }
  cond cnda (i=0..4; j=0..39) {        /* (3)条件，計算記述 */
    goukei[i][j] = seiseki[i][0][j]+seiseki[i][1][j]+seiseki[i][2][j];
  }                                    /* この例では 成績の合計を計算 */
  msg {                                /* (6)ヘルプ，エラーメッセージ記述部 */
    seiseki[ ][0][ ] = help: "国語のテストの得点を入力して下さい";
    seiseki[ ][1][ ] = help: "数学のテストの得点を入力して下さい";
    seiseki[ ][2][ ] = help: "英語のテストの得点を入力して下さい";
    seiseki[ ][ ][ ] = type: "整数値で入力して下さい",
                       range:"得点は0－100の間で入力して下さい";
  }
}

/* ③実際の入出力の記述（C言語による呼出部分） */
  sid=opensession(SIO,namelist,seiseki,goukei);   /* 入出力開始 */
  while(getstat(sid,ALL) != DONE);                /* 入出力終了を待つ */
  closesession(sid);                              /* 入出力終了 */
```

図 9 UIDL による記述例
Fig. 9 Example of UIDL description.

（4）入出力データごとのリトライ回数，
（5）入出力データが満足すべき条件を表す式，
（6）各種メッセージ
を記述する．(1)では，セッション名とそのセッションで入出力されるデータを表す仮の名前と構造をC言語の関数定義と同様の記述で定義する．図では，SIOという名前を持つセッションが定義されている．(2)は，画面上で二次元の書式や変換規則を表すテンプレートの指定とテンプレート内のフィールドと呼ばれる領域とデータの対応を示すもので，図では，tempaという名前のテンプレートが指定され，略記法を用いて簡潔にフィールドの割り当てが記述されている．また，変数 i が3次元データの切り口を表す切り口変数として用いられている．(3)～(6)の項目は，必要な場合のみ記述すればよく，図では(3)と(4)は記述されていない．さらに，(5)の条件式が代入に関しては常に真であるという性質を利用して合計得点の計算を行っている．また(3)，(4)，(6)の各項目は，単一データに対してだけでなく，データ構造の領域を指定して割り当て可能である．図では，メッセージの定義の部分で3次元配列 seiseki の2番目の添字を0に固定して，他の部分を指定しないことで“国語のテストの得点を入力してください”というメッセージを seiseki の2番目の添字が0であるすべての配列要素に割り当てている．全体として，この部分の記述は宣言的に行えるため，記述が容易で，また，理解しやすい．そのため，AP の UIF の設計変更等にも容易に対処できると考えられる．また，デフォルト値やメッセージの内容は，UIDL がインタプリタ方式で実行されるために，ユーザによる変更も可能である．

最後に，③は，幾つかのプリミティブを使ってCプログラム中に記述される．これらのプリミティブは，
（1）セッションの開始 (opensession)，
（2）セッションの状態の問い合わせ (getstat)，
（3）セッションの状態の再設定 (setstat)，
（4）セッションの終了 (closesession)，
の四つである．

（1）は具体的には，入出力するデータの実体をセッションに対応付け，ユーザに対してテンプレートを通しての入出力を開始する．(2)は，各入出力データまたはデータ全体に対して，入力が終了したか否か，出力が終了したか否か，現在までの入力エラー回数を得る．(3)は，逆にこれらの状態を設定する．(2)と(3)により，同一セッションでの再入力や，入力エラー時のより細かい処理が AP 側で可能であり，高度に対話的な AP の UIF も記述できる．(4)は，入出力を終了し，対応するテンプレートのユーザに対する表示を終える．図では，セッションを開始し，すべてのデータが入力されるまで待ち，その後セッションを終了している．

UIDL を用いて，文献カード管理システム等各種 AP の UIF の記述を行った結果，C言語に慣れたプログラマには，違和感がなく，また UIDL 部分で UIF に集中できるためユーザ本位の記述ができることが実際に経験できた．また，UIMS で提供される機能を実現するための記述を個別に行う場合よりははるかに少ない記述量で高度な UIF が記述できる．実際，図9の例とほぼ同等の機能を持つプログラムを言語Cで記述した場合約700行を必要とした．

5. システムの実現

本章では，UISE の実現のうち，UIDL コンパイラの処理と UIMS の実現アルゴリズムについて簡単に述べる．

5.1 UIDL コンパイラの処理

今回作成したC言語用 UIDL コンパイラの行う作業は大別して次の四つである．

① 入出力に用いられるデータ構造の解析
② セッション記述の処理
③ 入出力プリミティブの処理
④ C言語ソースプログラムの出力

①の処理は，実際にはどのデータ構造が入出力に用いられるかが，セッションの記述が行われるまでは確定されないために，ソースプログラム中のすべてのデータ構造を解析し記憶する．②の処理はセッション単位ごとに生成される UIDL データを出力する．③の処理は，ソースプログラム中に記述された入出力プリミティブを従来のC言語処理系で処理可能な形に変換する．具体的には，

（1）opensession 中のセッション名は文字列に変換し，

（2）入出力の変数を表す表現はその変数の先頭を表すアドレスに変換する．ただしすべての入出力変数を表す ALL は特殊な定数に変換する．

（3）opensession, getstat, setstat, closesession はそれぞれ UIDL ライブラリ関数として処理される．

例えば，図9の，

```
sid=opensession(SIO, namelist, seiseki,
```

goukei); while(getstat(sid, ALL)!=DONE);
という記述は,
sid=opensession("SIO", 3, namelist, seiseki,
goukei); while(getstat(sid, -1L)!=DONE);
と変換される.

④はソースプログラム中に含まれる入出力に関係のない部分をそのまま出力する処理である.

以上の処理を行う場合, C言語の構文をある程度解析する必要が生じる. そのためCコンパイラの構文解析部を変更し UIDL コンパイラの構文解析部として利用し, これにより処理系作成のコストを削減した.

UIDL コンパイラの生成する UIDL データは, 一つのセッション記述について, セッション名, 入出力データ表, テンプレート表, 条件表, メッセージ割り当て表, デフォルト値表の六つの部分から構成される. これらの表は AP の実行時に UIMS によって, 効率良く解釈実行されるよう設計した. これは, セッションでの UIMS の処理能力が, ユーザにとって UIMS の応答時間として感じられるためである. 以下では, 各部分について簡単に述べる.

① セッション名

後続する UIDL データが構成するセッションの名前を示す部分である. この部分には後続する各 UIDL データのサイズの合計も記述する.

② 入出力データ表

セッションで入出力されるデータの構造と入出力状態, 再試行回数, デフォルト値の存否を管理する表であり, セッションでの仮入出力引数ごとに, その仮引数の持つ固有の番号である仮引数番号 (仮引数としての出現順位) とサイズ, および仮入出力引数と同じ構造を持つ入出力管理フラグから構成される. 入出力されるデータと同じ構造を持つ入出力管理フラグを採用したことにより, テンプレート表から得られる入力データの変位から入力されたデータに対するフラグの取り出し, フラグから実体への変位の変換が容易に行える.

③ テンプレート表

セッションで使用するテンプレートの指定およびテンプレートのフィールドと仮入出力引数の割り当てを示す表から構成される. フィールドと仮入出力引数の割り当ては, テンプレート内のフィールド番号と仮引数番号および仮入出力引数内でのそのデータの先頭からの変位との組で示される. これにより, 構造体や配列のように一つの仮入出力引数に複数のデータの実体が含まれる場合に, これらを指定できる. さらに, ここで割り当てられる仮入出力引数が切り口変数を用いて表されている場合があるので, 変位は切り口変数による変位を計算する中間コード列として表す.

④ 条件表

UIDL の条件文を処理して生成される表であり, 条件判定を行うための中間コード列が条件式ごとに生成される. 条件式中に現れる仮入出力引数はテンプレート表の場合と同様に仮入出力引数番号と変位によって表される.

⑤ メッセージ割り当て表

条件エラー, その他のエラーで表示されるメッセージを集めた表で, 条件エラーのメッセージの場合は, 条件式とメッセージの割り当てが, その他のエラーでは仮入出力引数とメッセージの割り当てが出力される.

⑥ デフォルト値表

各仮入出力引数ごとにデフォルト値を集めたもので仮入出力引数ごとに仮入出力引数と同じ構造を持つデフォルト値表が用意される. したがって, 表中にはデフォルト値の存在しないデータに相当する部分も存在するが, デフォルト値の存否は入出力管理フラグで示されているので判定が可能である.

5.2 実現アルゴリズムの概要

本節では UIMS の実現のうち, UIDL データインタプリタの実現手法について述べる. AP が起動されると UIMS は UIDL データインタプリタを起動する, UIDL データインタプリタは, 次の手順で処理を行う.

① AP の起動に伴い必要な UIDL データファイルを読みこむ.

② セッションが開始される (opensession) と対応するセッションの中間コードを用意し, 必要なテンプレートデータファイルを用意しテンプレートをユーザに対して表示する.

③ 入出力管理フラグ表をもとに入出力されるデータのうち出力データを表示する.

④ あるフィールドが入力された場合 (デフォルト値の入力を含む) UIDL データインタプリタは, (1) そのフィールドの入力に対してテンプレートで指定されている変換を行い, (2) そのフィールドに対応するデータをテンプレート表をもとに割り出し, (3) 指定される範囲のテストを行い, (4) 各種エラーがあれば, これを表示し入出力管理フラグの再試行回数

をカウントし，エラーがなければ実際のデータに値を設定し入出力管理フラグを入力済みに設定する，(5) 入力されたデータに関係する条件式があれば，これを計算し，条件が成立しない場合は，条件エラーのメッセージを表示する．

⑤ AP から入出力管理フラグの問合せおよび再設定が行われればこれを行い，入力済みのフィールドが未入力に設定された場合は，画面表示を入力前の状態に，出力済みのフィールドが未出力に設定された場合は，再度出力を行う．

⑥ ユーザが入力終了キーを押した場合は，すべての入出力変数に関して未入力の部分がないか調べ，未入力の部分があれば，その部分へカーソルを移動し入力を促進する．未入力の部分がなければ，次にすべての条件式の成立を調べ，成立しない条件があれば，メッセージを表示する．条件も成立した場合には，入力完了の状態となる．

⑦ AP からセッションの終了要求 (closesession) が来れば，画面の表示を終了し，このセッションに対する処理を終了する．

これらの処理のうち，テンプレートのフィールドから入出力変数のデータ格納場所への変換はテンプレート表から容易に可能である．しかし，あるデータに割り当てられたメッセージ，そのデータを引数とする条件式を選択するには全メッセージあるいは全条件式について該当する変数が割り当てられているか調べる必要がある．UIDL による記述ではメッセージは入出力変数全体だけでなく，構造を持った入出力変数の一部に割り当てることも可能であるため，メッセージの割り当てられている入出力変数中の領域が不連続になる．簡単にはこの領域を表す変位の範囲を表にしておき，割り当てを調べるデータの変位がこの範囲に入っているかを調べればよいが，この方法では，メッセージの割り当てられた領域の個数が増えると比較回数が増加することで，ユーザに対して応答時間の遅れが生じ，好ましくない．そこで，一般に，n 次元の配列の要素 $a[x_n][x_{n-1}]\cdots[x_2][x_1]$ の変位 O は各次元の添字を x_i とすると，

$$\sum_{i=1}^{n} C_i x_i, \text{ ただし } C_i \text{ は定数}$$

で表されるので，

ある変位 O が与えられた場合，これを導出する $x_1\cdots x_n$ は，

$C_i x_i > \sum_{j=1}^{i-1} C_j x_j$ という性質より，

```
for i := n to 1 do begin
  x_i := O div C_i;
  O := x_i mod C_i
end
```

で求められることを利用して，変位の範囲の一致を調べるのでなく，変位を生成する各添字を求め，これの範囲の一致を調べることとした．入出力変数が構造体を含む場合も変位を表す一次式に定数項が付加されるだけであり，同じ手法を用いることができる．この方法を用いることにより要素数の大きさに左右されずに，配列の次元，または構造体のネストの深さにのみ比例した時間でメッセージの割り当てを調べることができる．条件式の処理についても同様の判定法を用いた．この手法により，UIMS でのセッションの処理を高速化でき，UIMS のユーザに対する応答を，セッションで扱うデータ量が多い場合でも，高速にできる．

6. あとがき

現在，UISE のプロトタイプの作成を行い，既に UIDL コンパイラなどが実際に試用されている．

本システムで用いた実現手法は，ユーザにセッション内での大きな自由度を与えることが可能である．また，プログラマにとっても，UISE は，広範な AP において，UIDL を用いて比較的少ない記述で，統一のとれた比較的良好な UIF を提供することを可能にする．このため，本システムはプログラマにもユーザにも十分利用価値を持つと考えられる．

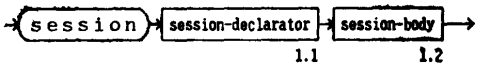
謝辞　本システムの UIDL コンパイラの実現に御協力いただいた今中崇雄氏（現シャープ株式会社）に感謝いたします．

参考文献

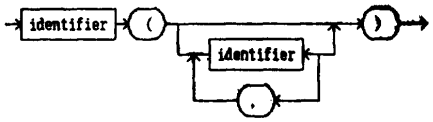
1) Hammer, M., Kunin, J. S. and Schoichet, S.: What Makes a Good User Interface?, *1983 Office Automation Conference Digest*, AFIPS Press, pp. 121-130 (Feb. 1983).
2) Bass, L. J.: A Generalized User Interface for Applications Programs (II), *Comm. ACM*, Vol. 28, No. 6, pp. 617-627 (1985).
3) 竹村，辻野，荒木，都倉：ユーザインタフェイス作成支援システムについて，情報処理学会ソフトウェア工学研究会資料，46-19 (Feb. 1986).

付録 UIDL 構文図（概要）

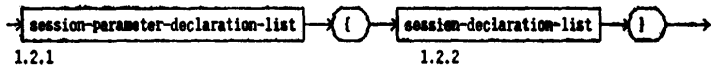
1. session-definition



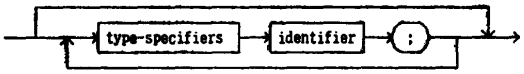
1. 1 session-declarator



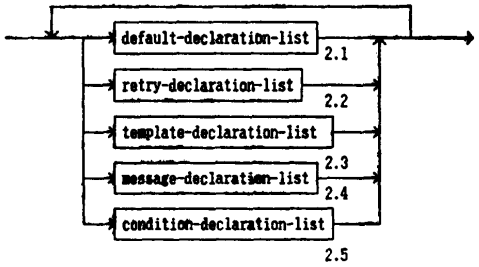
1. 2 session-body



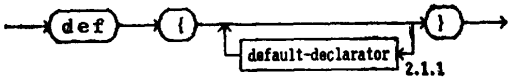
1. 2. 1 session-parameter-declaration-list



1. 2. 2 session-declaration-list

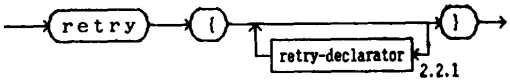


2. 1 default-declaration-list

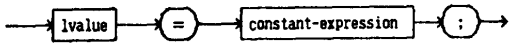


2. 1. 1 default-declarator

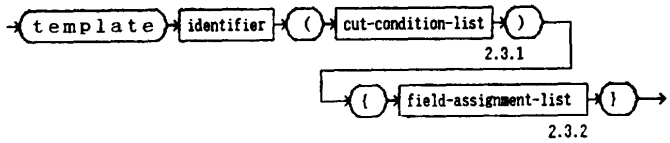
2. 2 retry-declaration-list



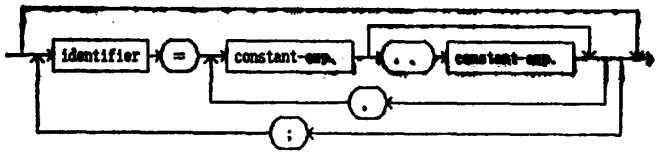
2. 2. 1 retry-declarator



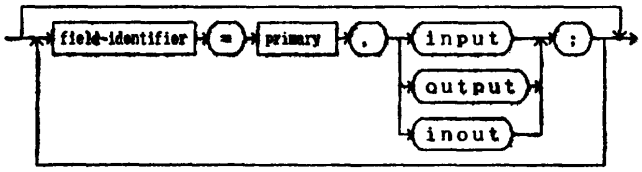
2. 3 template-declaration-list



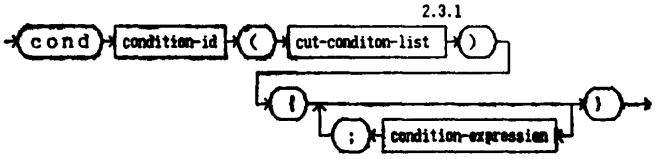
2.3.1 cut-condition-list



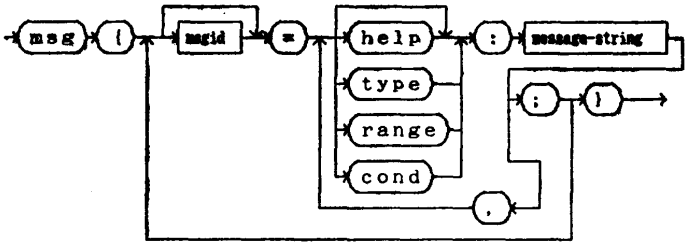
2.3.2 field-assignment-list



2.4 condition-declaration-list



2.5 message-declaration-list



(昭和61年4月1日受付)
(昭和62年3月25日採録)

竹村　治雄（正会員）

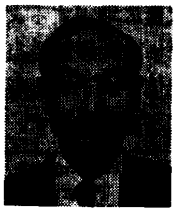
昭和57年大阪大学基礎工学部情報卒業．昭和62年同大学院博士課程単位取得退学．同年国際電気通信基礎技術研究所入社．主として，計算機のユーザインタフェースに関する研究に従事．電子情報通信学会会員．

辻野　嘉宏（正会員）

昭和54年大阪大学基礎工学部情報卒業．昭和59年同大学院博士課程修了．同年同大学基礎工学部助手．工学博士．計算機言語，並行処理記述，プログラミング教育，ユーザインタフェースに関する研究に従事．電子情報通信学会会員．

荒木　俊郎（正会員）

昭和48年大阪大学基礎工学部制御卒業．昭和51年同大学院博士課程（情報）中退．同年大阪大学基礎工学部情報助手．昭和54年講師．現在，助教授．工学博士．主として，オートマトンと言語理論，プログラムの理論と技法に関する研究に従事．電子情報通信学会会員．



都倉　信樹（正会員）

昭和38年大阪大学工学部電子卒業．昭和43年同大学院博士課程修了．工学博士．同年同大学基礎工学部講師．現在，同教授（情報工学科）．主として，プログラムの技法と理論，計算機言語，VLSIの計算複雑さの理論などの研究に従事．ACM，電子情報通信学会，日本ソフトウェア科学会各会員．