4R−3　ユーザインタフェース設計における概念記述の変換

三宅一巧，渡辺喜道，今宮淳美
山梨大学

1．はじめに

従来のユーザインタフェース管理システム(UIMS)[3]では対話形式の設計を支援するが，対話の意味の設計は支援していない．

そこで本研究では，Foleyが提案する4レベルユーザインタフェース設計[1]における概念レベルの設計をサポートするため，ユーザインタフェース定義言語IDL(Interface Definition Language)[2]で記述した概念設計の仕様記述を機能的に等価な複数の異なる仕様記述に変換するシステムを開発する．

2．概念記述の変換

2．1　概念記述

4レベルユーザインタフェース設計において最初に行われる概念設計は，オブジェクトとその属性や関係，オブジェクトに対するアクションとそのパラメータ，前条件，後条件などを定義する．本研究ではこれらの定義をIDLにより記述し，これを概念記述と呼ぶ．

```
{1．オブジェクトの階層構造の定義}
type shape
    superclasses     :()
    subclasses       :(triangle,square)
    actions          :(create_shape,delete_shape,rotate_shape):
                      originates,heritable
    attributes       :(position,color):originates,heritable
{2．属性値の範囲の指定}
position:range[0.,10]x[0.,10]
color:set(1,1)of(red,blue,green,white,cyan,magenta,yellow)
angle:range[0.,360]
{3．前条件の初期値設定}
initial:number_shape=0
{4．アクションの定義}
create_shape  (p:position,c:color,a:angle,type_of_shape:shape)
postcondition (number_shape=number_shape+1)

precondition  (number_shape<>0)
rotate_shape  (obj:shape,a:angle)

precondition  (number_shape<>0)
delete_shape  (obj:shape)
postcondition (number_shape=number_shape-1)
```

図1．概念記述の例（一部）

2．2　変換の有効性

概念記述は同じ機能を持つユーザインタフェースに対して様々な仕様があり，設計者はこれらについて使いやすさ，使用時のスピードなどを考慮する必要がある．このような考慮は，ユーザインタフェースが複雑になれば非常に面倒な作業である．

そこで本研究では，1つの概念記述に自動的に様々な変換を施し，機能的には等価な複数の異なる概念記述を生成するシステムを開発する．このシステムを利用して様々なパターンの概念記述を生成することにより，各インタフェース使用時のスピードなどを評価し，適切と思われる記述を現存するUIMSへ入力することで，簡単で素早いユーザインタフェース構築が期待できる．（図2）

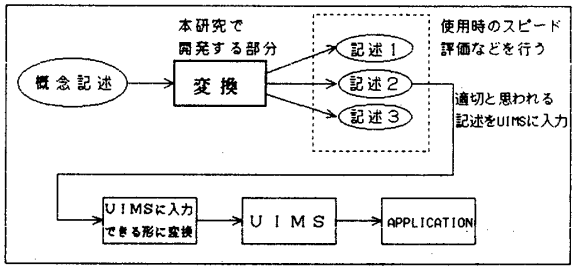

図2．概念記述変換を用いるユーザインタフェースの構築

2．3　変換の例

CSO(Currently Selected Object)やCSC(Currently Selected Command)の概念を用い，コマンド実行時の手間を少なくするための変換が因子化である．すなわち，すべての操作をCSOに対して行うことによりコマンド実行時のオブジェクト指定の必要をなくしたり，CSCの概念を用いて繰り返し同一コマンドを指定する必要をなくす．

3．変換システム

3．1　変換システムの機能

本研究で作成した変換システムは次の4機能

Transformations on a Conceptual Specification of User Interface Design
Kazuyoshi MIYAKE, Yoshimichi WATANABE, and Atsumi IMAMIYA
Yamanashi University

を持つ．

(1)概念記述のエラー検出
(2)無駄な記述の検出
(3)概念記述の変換
(4)概念記述ファイルの操作

変換の種類には，オブジェクト属性の因子化，オブジェクトの因子化，コマンドの因子化の３つがある．また，概念記述ファイルの操作には，作成，削除，編集，コピーなどがある．これらの操作をすべてメニュー形式のユーザインタフェースで実行できる．（図３）

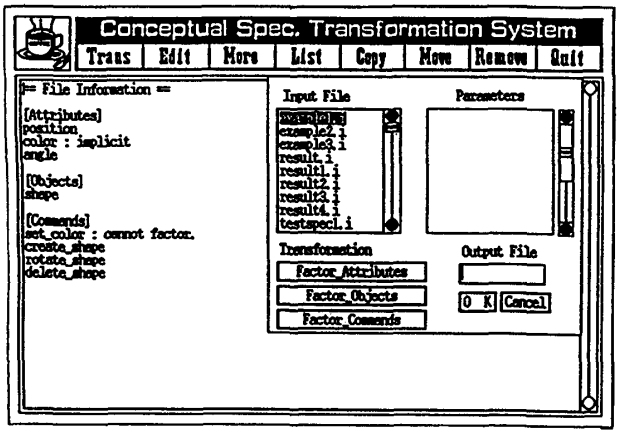

図３．変換システムの画面例

３．２　変換システムの実現

図４に変換システムの構成を示す．また，変換システム中の実際に変換を行う部分は図５に示すように，語彙解析部，構文解析部，意味解析部，変換部の４つの部分からなる．このプログラムではトークンが生成されるたびにその値をファイルに出力する形式を用いたため，解析している部分より前の部分の変換は不可能である．そこで，構文解析部で変換できる部分は変換を施し，その結果を中間ファイルに出力する．そして意味解析部で変換を施すべき位置や，そこをどう変えるかなどの変換に関する情報を抽出し，変換部でそれら変換情報に従い中間ファイルを参照しながら，さらに出力ファイルに書き出すという方法を用いる．

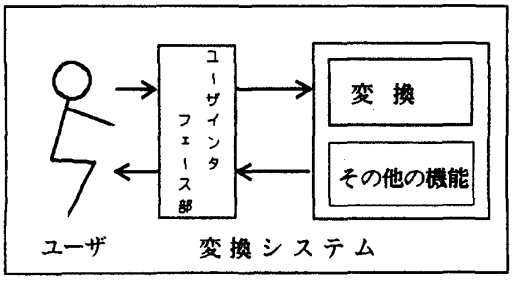

図４．変換システムの構成

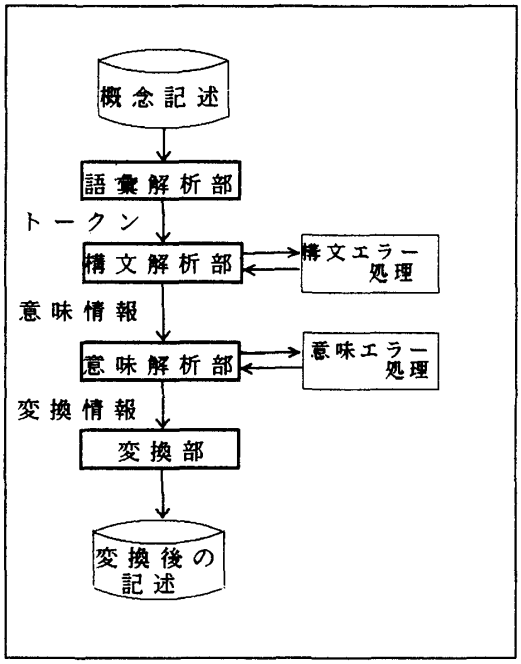

図５．変換部の構成

４．おわりに

本研究では，概念記述を機能的には等しいが仕様の異なる複数の概念記述に変換するシステムを作成した．

現在実現している変換の種類は３種類でありこれでは十分と言えないので，今後新しい変換アルゴリズムを開発してこのシステムを強化する予定である．また現在，概念記述ファイルを設計者がテキストエディタにより作成しなければならない．そのため概念記述を対話型で作成可能とするツールを開発し，容易に記述作成ができることが必要である．本研究では４段階の設計での概念設計のみを対象としたが，その他の部分の支援も行えるようにしたい．

参考文献

[1] Foley, J.: Models and tools for the designers of user-computer interfaces, Report GWU-IIST-87-03, Dept. of EE&CS, George Washington University, 1987.

[2] Gibbs, C., W.C.Kim, and Foley, J.: Case studies in the use of IDL: Interface Definition Language. Report GWU-IIST-86-30, Dept. of EE&CS, George Washington University, 1986.

[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，オーム社，pp. 1194-1201, 1989.