4M-5

ユーザインタフェース構築支援システム -メタユーザインタフェース構築支援-

荒井　俊史　　谷越　浩一郎　　谷　正之
(日立製作所　日立研究所)

1．はじめに

ユーザインタフェース構築支援システム-MU-について報告する。従来より，ユーザインタフェース(UI)の構築支援のため，UI管理システム(UIMS)を開発してきた [1-3]。これにより，エンドユーザが使うUI (目的UI)の構築は効率化できた。今回開発したMUは，目的UIの構築支援だけでなく，目的UIをカスタマイズするためのUI(メタUI)の構築も支援できることを特徴とする。

2．メタUI構築支援の必要性

計算機の利用者/形態が多様化するにしたがい，エンドユーザによるUIのカスタマイズが重要になっている。マクロ機能などもカスタマイズの一例であり，CADやスプレッドシートなどでは重要な機能である。また，ハイパーテキストなどでは，テキスト間のリンクを定義するボタンやフィールドを生成する手段が備わっている。メタUIとは上例ように，プログラムの実行中に対話的に簡単にUIをカスタマイズする手段であり，いろいろな分野で必要不可欠な機能になりつつある。

従来のUIMSは，UIを構築/カスタマイズするための汎用的な手段を提供することを目指していた。しかし，CADシステムとスプレッドシートでは，システムの特徴や利用者の傾向が異なるため，UIをカスタマイズする方法も異なったものが必要であろう。すなわちUIMSは，適用分野に応じて，適切なメタUIを提供できることが必要である。

また，UIMSが標準で提供する機能(部品)で機能不足が生じた場合，ユーザが独自に部品を追加することになる。この場合，追加した部品を編集するためのUIが必要となる(すなわちメタUIの拡張が必要となる)。この場合も，メタUIがシステムに作り付けになっていると，部品を追加するたびにUIMS自身を手直しする必要が生じる。このような手間を省くためにも，メタUIを構築/編集できることが必要である。

3．メタUI構築支援機能(MUの機能)

MUは図1のような構成になっている。各構成要素(部品)の動作を実行するカーネルがあり，UIはカーネルが実行するデータとして実現する。よってUIの定義データを追加/変更することで，UIを構築/編集できる。部品の動作定義データは，ルールから生成される。例えば，マウスボタンが押された時に応用プログラムの関数を呼び出すには，UIDL(UI定義言語)で図2のようなルールを書けばよい。また機能部品 (メニューやアイコンなど)の定義データを生成する時は，類似の機能を持つ部品をプロトタイプとし，その性質 (表示形態やルールなど)を継承したものを生成できる。ルールは個々の部品ごとに必要に応じて

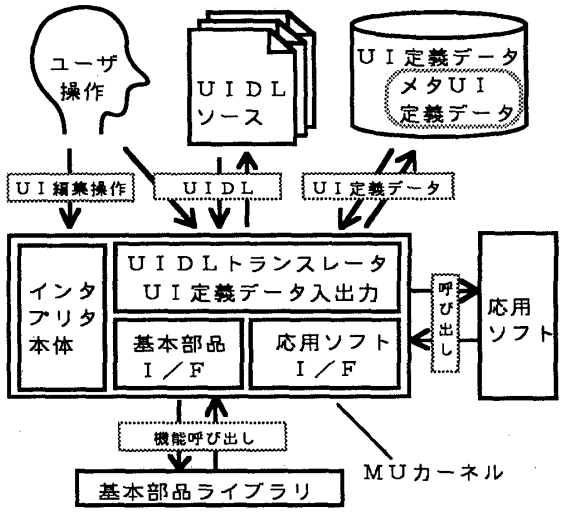

図1 MUの全体構成

A System for Developing User Interface
- Meta User Interface Support -
Toshifumi ARAI, Kouichirou TANIKOSHI, Masayuki TANI
Hitachi Research Laboratory, Hitachi, Ltd.

再定義できる。

一方，メタＵＩをカスタマイズ可能とするには，目的ＵＩと同様に，定義データの形で実現すればよい。すなわち，メタＵＩをＵＩ定義データを追加/更新する機能を持つデータとして実現する。メタＵＩの定義データはＵＩＤＬ(ＵＩ定義言語)のソースでカーネルに与え，目的ＵＩの定義データはメタＵＩの機能またはＵＩＤＬで生成/定義する。

しかし，メタＵＩと目的ＵＩを同じ枠組みで構築すると，使える入力装置と装置への操作が限られているため，同じユーザ操作に複数の処理を対応させる必要が生じる。例えばメニューに対するマウスのドラッギングが，メニューの項目選択(目的ＵＩ)とメニューの移動(メタＵＩ）の２通りに使われるという場合である。

このような処理を素直に表現すると，図３のようなルールになる。しかし，この実現では次のような問題がある。すなわち，目的ＵＩのカスタマイズに必要なのはルール中の枠で囲んだ部分だけであるのに，その部分を修正するために他の部分を知らなければいけない。これでは修正の効率が悪いし，誤りも起こりやすい。また，メタＵＩのための定義と目的ＵＩのための定義が混在しているため，メタＵＩに関する定義はプロトタイプから継承したものを用い，目的ＵＩに関する定義だけを再定義するといったことができない。すなわち，メタＵＩを目的ＵＩと同じ枠組みで構築すると，動作の定義/編集が困難になってしまうという問題がある。

ＭＵでは上記の問題を解決するため，マウスやキーボードなどの入力装置を仮想化する手段を設けた。例えば，目的ＵＩを実行するためのマウスの他に，部品移動用マウスがあると想定して動作を定義できる。この仮想化はイベント名称を変更する機構で実現した。目的ＵＩとメタＵＩで異なる入力装置(イベント名称)を使うと想定すれば，同じユーザ操作に対する部品の動作でも，独立したルールで定義できる。例えばマウスのドラッギングに対して，目的ＵＩのために drag，メタＵＩのために drag_to_move というイベント名称を使うことにすれば，図３は図４の２つのルールに書き換えられる。イベント名称の切り換えは，ルールの処理部分に指定できる。例えば，あるアイコンに図５のようなルールを定義すれば，それをクリックした時にイベント名称を切り換えることが可能となる。ＭＵのカーネルは，発生したイベントの名称により，適切なルールを選択/実行する。

```
event hold { call AP_func() }
```

図２ 動作定義の例（ルール）

```
event hold {
    if (mode == META) {
        move_by_drag()
    } else {
        select_item()
    }
}
```

図３ メタＵＩと目的ＵＩが混在したルール

```
event drag { select_item() }
event drag_to_move { select_item() }
```

図４ 入力装置を仮想化した場合のルール

```
event hold {
    use_event("hold_to_move","drag_to_move")
}
event hold_to_move {
    use_event("hold","drag")
}
```

図５ イベント名称を切り換えるルール

図４のように，異なるイベント名称に対して定義したルールは，互いに独立しているから，目的ＵＩに関する部分だけを容易に修正/再定義できる。

４．おわりに

目的ＵＩとメタＵＩの両方を構築支援できるＵＩＭＳ -ＭＵ- を開発した。メタＵＩは，ますます重要になるものと考えられる。今後は，ＭＵをいろいろなシステムに適用し，メタＵＩ構築支援機能を検証することを予定している。

参考文献

[1] 谷，横山，荒井：ユーザインタフェース構築支援システム(1)-開発構想-

[2] 横山，谷，荒井：ユーザインタフェース構築支援システム(2)-開発環境-

[3] 荒井，谷，横山：ユーザインタフェース構築支援システム(3)-実現方式-

以上，情報処理学会第35回全国大会 4W-2 ～ 4W-4