EWS4800 シリーズ

# 4M-3 — ユーザインタフェース構築基盤システム: 鼎(かなえ) —

暦本純一, 菅井 勝, 山崎 剛*, 森 岳志,
内山厚子, 垂水浩幸, 杉山高弘, 秋口忠三

日本電気(株) ソフトウェア生産技術開発本部,
* 日本電気マイコンテクノロジー(株)

## 1 はじめに

最近の著しいハードウェアの機能向上と低価格化, X-Windowのような共通基盤としてのウインドウシステムの普及によって, マルチウインドウ環境化で, テキストだけでなく各種のメディアを扱えるようなアプリケーションへの要求が非常に高まってきている. ところが, これらのアプリケーションでは, ユーザとの対話部分(以下UI部と略す)の実現に高度なプログラミングを必要とし, UI部の開発がネックになってシステムの構築を困難なものにしていた. UI部は, 実際にそのシステムを使いながら修正して, 使い易さを改善していくべきものであるが, 従来はそれも困難であった.

この問題を解決するために, アプリケーションからUI部を分離して, 専門のモジュールないしはプロセスに担当させようというアプローチ (UIMS – User Interface Management System) がある. 従来のUIMSではUI部の対話制御(メニューからコマンドを選択する, 等)を担当し, 後はアプリケーションが担当することになっていた. しかしUI部の中で最も作成にコストがかかるのは, 画面上の対象物の編集操作を制御する部分である. たとえばCASEツールではモジュール階層図をユーザに編集させたり, モジュールの機能概要を日本語入力させる部分の構築に手間がかかる. 逆に, 多くのアプリケーションを編集という観点から見直すと, 目的は異なっていても共通するものが多い. 編集機能の共通的な部分を部品化し, アプリケーションごとの差異を吸収する拡張機能をUIMSが持てば, UI部の作成コストは大幅に減少すると予測される.

このような発想に基づいて, われわれはUI部の構築基盤システム「鼎(かなえ)」を開発している[2]. 鼎では, 各種のアプリケーションで共通に使われる編集対象の種類(メディアタイプ)を6種類(**テキスト, 表, 図形, グラフ構造, 階層構造, イメージ**)選定し, その編集機能(ファイル入出力, 画面表示, マウス操作, キー操作等)を部品化している. アプリケーション固有の処理は, ユーザがキーやマウスを操作したときに呼びだされるコマンドを, 専用のスクリプト言語(Lispベースのインタープリティブな言語)によって再定義することによって, 自由に変更できるようにした. 図1に鼎の上に構築されたツールの画面例を示す.

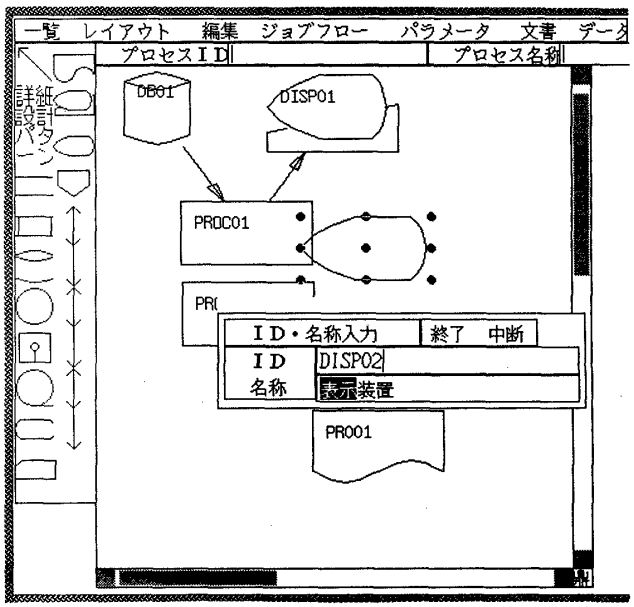


図 1: 鼎の画面イメージ

## 2 鼎のシステム構成

図2に鼎の全体構造を示す. 鼎はNEC EWS4800シリーズ等のUNIXワークステーション上で, X-Windowのクライアントとして動作するひとつのUNIXプロセスである. 実現にはX-Toolkit[1]を利用している.

**対話部品**は, 対話処理をするための基本的な部品群で, ボタン, メニュー, リスト(スクロールするメニュー), ボリューム(スクロールバー), パレット(図形からなるメニュー), フィールド(1行入力)が用意されている.

**台紙**は, アプリケーションが開くウインドゥに対応した概念で, 台紙の上に置かれている対話部品, エディタ部品の位置やサイズ, 対話部品やエディタ部品から起動されるスクリプトプログラムを保持している.

**スクリプトプログラム**は, エディタ部品が提供する基本的な機能を組み合わせて, アプリケーション向けの機能を提供するための言語であり, (1) キーボード入力,

Canae: A Platform of User-Interface Developement, *Junichi Rekimoto, Masaru Sugai, Go Yamazaki*, Takeshi Mori, Atsuko Uchiyama, Hiroyuki Tarumi, Takahiro Sugiyama, Chuzo Akiguchi*, NEC Corporation, * NEC Microcomputer Technology, Ltd.

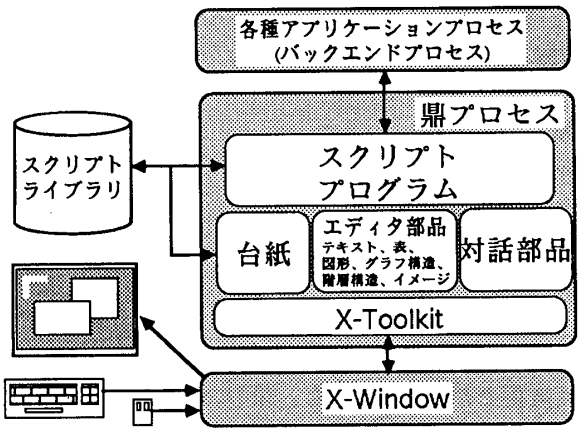


図 2: 鼎のシステム構成

(2) 対話部品のマウスによる操作, (3) エディタ部品の画面上でのマウスによる操作, (4) バックエンドプロセスからの送信, といった事象に応じて対応するスクリプトが起動される.

アプリケーションのUI部以外の処理は, アプリケーションごとに作成される別プロセス (**バックエンドプロセス**と呼ぶ) が担当する. バックエンドプロセスは, 新規開発したものでもいいし, 従来のUNIXコマンドでもいい. 後者の場合, 既存のツールに対して, 鼎を視覚的なUIを持つフロントエンドとして使う方法になっている. 簡単なアプリケーションの場合は, すべてをスクリプト言語で書き切ってしまうことも可能である.

以上のような機構を用いて, モジュール構成を操作するツールを鼎の上に構築する場合を考えてみる. モジュール構成の表現にはグラフ構造を使うことにする. グラフ構造上のノードと実際のモジュールとの対応は, ノードに添付する属性によって表現する. ノードを表示させたり, 移動させたりする機能は, グラフ構造エディタ部品の機能がそのまま利用できる. ノードとノードをアークでつないだときに, その関係をモジュールのデータベースに追加する機能は, ツールに特有のものなので, スクリプトによって記述する.

このように, 新しいアプリケーションを作成するときは, それに特有の部分だけを拡張言語によって記述すればよく, 従来のように全てを作りあげる方針と比較して, 大幅に作成コストが減少する.

## 3 鼎のエディタ部品の構成

### 3.1 モデルとビューの分離

鼎のエディタ部品は, ユーザインターフェース構築のためのモデルであるMVC (Model –View – Controller) **モデル**をエディタ向きに改造したものを用いている. 鼎のエディタでは, 連続するモデルの変化に対しては変化情報をバッファリングし, ビューが画面を更新する回数を必要最小限におさえるように工夫している.

エディタ部品をモデルとビューを明確に分離しているため, ひとつのモデルに複数のビューを設定すること, ひとつのモデルに異なったタイプのビューを設定すること, が極めて容易に実現できた. 前者は編集対象の複数の箇所を別々の画面から操作するときに, 後者は精密表示と概略表示のように, 編集対象の表示方式がいくつも考えられるときに必要となる機能である.

### 3.2 属性の添付

鼎のエディタ部品では, 扱っているモデルの基本要素 (選択できる最小単位) ごとに, 属性情報を格納する枠組を用意している. 属性情報はタグつきの文字列データであり, その使用方法はアプリケーションが自由に決定できる. 属性として格納するアプリケーション依存情報としては, (1) スクリプトプログラム, (2) ファイル名 (パスネーム) (3) データベースの検索キーなどがある. (2) はひとつの文書から別の文書へのつなぎ情報 (ハイパーメディアリンク) を実現するための基本的なメカニズムになっている. (3) は, つなぎ情報がデータベースに格納されているような場合のリンクの実現方法として使える.

### 3.3 異種メディアの混在

あるメディアの中への別のメディアの貼り込み, 異なるメディア間でのデータ交換を可能にするために, メディアの**フラグメント**という概念を導入している. フラグメントはバイトストリーム形式のデータで, **殻**と**中身**からなっている. 中身は, 各メディア (モデル) をバイトストリーム形式に変換したものであり, 殻に変換したメディアのタイプと表示関数, フラグメントを表示したときの画面上での大きさが記述されている.

フラグメントを他のメディアに貼り込んでいるときは, 殻に記述されている情報のみを利用し, 中身のデータには立ち入らないようになっている. したがって, 任意のメディアを別のメディアに自由に貼り込むことができる. あるメディアに対するフラグメントを実現するのに, メディアごとに必要となる作業は, フラグメントとモデルの相互変換関数と, フラグメントの表示関数を実装することだけである.

## 4 おわりに

マルチメディアを扱うアプリケーションの構築基盤となるシステムを設計し, 現在実装中である. 今後は機能の充実をはかると共に, 実際のアプリケーションによって, その実用性を評価していきたい.

## 参考文献

[1] Joel McCormack *et al.*, "*X Toolkit Intrinsics — C Language X Interface*", X Window System, Version 11, Release 2.

[2] 暦本, 菅井, 他「X ウィンドゥ上のマルチメディアユーザインタフェース構築環境: 鼎」情報処理学会第 30 回プログラミングシンポジウム予稿集, 1989.