2AE－4

階層構造を持つデータの入力システムのための アプリケーション・ビルダーの試作

広瀬 紳一
日本アイ・ビー・エム株式会社 東京基礎研究所

1. はじめに

データ入力のためのアプリケーションにおいて、入力・編集するデータの単位がデータベースの１レコードに対応するようなものであれば、ユーザーインターフェースは、一般には固定的なパネルで十分であり、現存するツールを利用して、非常に容易に作成することが可能である。しかし、もう少し複雑な構造を持ち、要素の個数が実行中に変化するようなリストを含むようなデータを扱おうとすると、階層の段階ごとに、リストボックスを用いて、対象を選択していく等の操作が必要となり、データの全体を見ながら入力を行なうことが難しくなる。逆に、いつでも全体を表示しておくためには、対象となるデータの変化に応じて、表示・編集用のコントロールを動的に生成・消去することが必要となり、開発のワークロードの増大につながってくる。

このような問題を解決するため、我々は、データ構造の定義を行なうだけで、その編集に必要なコントロール群からなるパネルを自動的に生成することを可能にするためのアプリケーション・フレームワークをC++を用いて開発してきた。本発表では、このフレームワークに基づいたアプリケーション開発に必要となるコーディングの作業量を、さらに削減することを目的として試作したアプリケーション・ビルダーについて述べる。

2. フレームワークの概要

我々のフレームワークは、入力・編集の対象となるデータを、基本的な要素が木構造をなすものとし、これを構成するためのクラスを提供する。いちばん下のノードには、整数、文字列といった、具体的なデータ要素がおかれ、上位のノードはそれらをグループ化するためのリスト等のクラスのインスタンスである(図１)。このような構造を持ったデータ(「モデル」と呼ぶ)の表示を行なうためには、木の根になっているオブジェクトの表示を、アプリケーションのメイン・ウィンドウ(に対応するグローバルなオブジェクト)に依頼すればよい。このとき、モデルの各要素に１対１で対応する「ビュー」と呼ばれるカテゴリーのオブジェクト群がフレームワークによって自動的に生成され、さらに、それらのビュー・オブジェクトが必要なコントロールを作成し、もとのモデルと関連付けることによって、画面上への表示が行なわれる。また、このビューは、担当するコントロールに対するユーザーの操作の結果をモデルの状態に反映させる、という役割も持っている。モデルとビューのクラスの対応は、必要に応じて、宣言的に記述しておくようになっている。

ここで、親モデル・オブジェクトに、新しい子オブジェクトが付け加えられたとする。親オブジェクトは、そのオブジェクトへのポインターを自分の管理するリストに登録するとともに、自分のビューに対して、自身の構成に変更があったことを通知する。ビューは、この通知を受け取ると、モデルを検査して、新たに加わったモデルに対応するビューを生成し、自分の子オブジェクトとして登録する。さらに、このことによって、実際のコントロールも作成され、画面上にもあらたな表示域ができる。反対に、木構造からモデルの一部が切り離された場合にも、ビューへ同様の通知がなされるため、対応するビューの子オブジェクトと、それに伴うコントロールの消去が起きる。このようにして、各アプリケーションでは必要に応じてモデル・オブジェクトの変更を行なうだけで、画面上の入力パネルの自動的な更新が行なわれることになる。

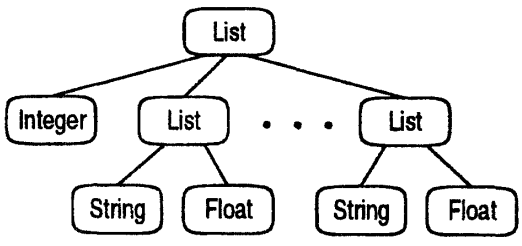

図１．フレームワークの扱うデータ構造

An Application Builder for Data Entry Systems for Hierarchically Structured Data

Shin-ichi Hirose
Tokyo Research Laboratory, IBM Japan, Ltd.

3. アプリケーション・ビルダーとその機能

このようなフレームワークと付随するクラス・ライブラリーを利用して実用的なアプリケーションを開発するには、C++によるプログラミングが必須となる。さらに、GUI画面の設計については、開発が終了した後にも様々な変更要求が発生することが多い。これらの状況に対処するため、我々はC++でのコーディングなしで、このフレームワークに基づいた開発のかなりの部分を行なうことを可能にするためのツールを開発しようとしている。その実現可能性を検証するために、GUI上でアプリケーション固有のクラスの定義や、それらとビューとの対応付けを行なうことを可能にする、簡単なアプリケーション・ビルダーを試作した。これは、与えられた定義を用いて、そのクラスの振る舞いを、その場でエミュレートしたり、ソースコードを生成したりする機能を持っている。

このアプリケーション・ビルダーについての特徴的な点のひとつは、それが対象としている「アプリケーションの定義」自体が、「動的に変更される階層的なデータ」である、ということである。すなわち、このビルダーは、我々のフレームワークの上に構築されているのである。図2の画面に示すように、Projectクラスのインスタンスの下に、メニューやツールバー、モデルのクラス、および、任意の数のアプリケーション固有のモデルのクラスの定義があり、これらのメンバーを定義していくことが、アプリケーションの記述に相当する。

基本的な操作としては、左側のペインで項目を選ぶことにより、右側のペインにその設定を行なうための、ノートブックが表示されるので、その中の各フィールドに適当な値を与えていけばよい。その後、preview コマンドを実行すると、動作の検証のための疑似的なインスタンスがその場で生成される。また、generate source コマンドを実行すれば、定義したクラスのソースコードが生成される。

その場での動作確認においては、当然、実際に新しいクラスを作ることはできないため、定義されたクラスは、継承関係をさかのぼって到達したフレームワークのクラスのインスタンスで代用される。ただし、ベースクラスがフレームワークの一般的なリストのクラスであった場合には、その、「実行時に自由に子ノードを付けはずしができる」という特性を利用して、定義されているメンバーもモデルとして付加するようになっている。また、各メンバーの属性として、「与えられたクラスのオブジェクトの配列やリストである」ことを宣言するためのものがあり、配列ならば、与えられた個数のインスタンスが付加されることになる。リスト属性を持っている場合には、親オブジェクトが生成される時点では子オブジェクトは生成されない。そのかわりに、子オブジェクトを後で生成して追加するためのメソッドが、そのクラスに自動的に定義される。このメソッドの動作は、プレビューを行なっている時にもエミュレートされ、追加されるコントロールが予期された位置に配置されること等を確認することが可能になっている。

4. おわりに

現在のプロトタイプは、GUI関係の動作のみを対象としており、その範囲においては、十分に実用的なものを作っていくことができることを試作を通して確認できた。しかし、現状のように種々のパラメーターをテキストのみで指定するのでは、まだ不十分であり、配置情報などはできるかぎりGUI上での操作で指定できるようにする必要があると考えている。今後は、このプロトタイプをもとに、フレームワークの提供するデータ編集機能のサポートや、モデルとデータベースとの接続を定義するための機能を追加し、実際のアプリケーション開発を通しての評価を行なっていく予定である。

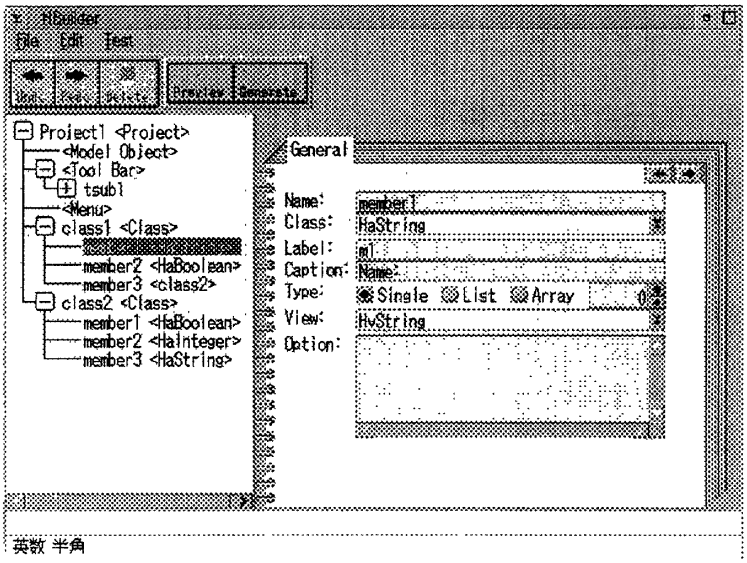

図2．アプリケーション・ビルダーの編集画面