7C−5

グラフ構造をもつデータベースのブラウザを作るためのフレームワーク

加藤 晶子† 三ッ井 欽一†
†日本アイ・ビー・エム(株)東京基礎研究所

1 はじめに

グラフ構造をもつデータベースのブラウザをC++で簡単に作るためのフレームワークをC++ライブラリとして提供することを目標にする．この Library（以下GvLibraryと呼ぶ）は，X Widow システム上でMotifに基づいたGUIを持つアプリケーションを作るためのものである．この Library のユーザーが，対象とするデータベースに固有な部分のみつくればアプリケーションができるように設計する．我々が具体的に考えているアプリケーションはプログラム構造図を表示するブラウザであるが，Unixのファイルシステムの構造図，会社と従業員の雇用関係図など，一般的にグラフ構造をもつもの全てを対象とする．また，表示だけでなく，GUIの画面上からデータベースにアクセスしてデータを変更する機能も合わせて提供する．なお，Motifベースの GUI Library は別に提供されていて GvLibrary はそのうえに作られている．

2 GvLibrary の提供する機能

GvLibrary は次の機能を提供する．

1. MVC モデル [1] を用いて，表示画面上からデータベースのグラフ構造を変化させたり，データベースのグラフを構成する各要素の状態・性質の変化させる機能．また，データベース側の変更を表示画面上に反映させる機能．

2. MVC の View 側のグラフ全体の表示位置の移動，zooming，レイアウト戦略の切替え，overview ウィンドウの表示の機能．また，View 側グラフの頂点や枝の表示する色の変更，線の太さや種類の切替えなど，グラフ表示上のコマンド群．また，頂点・枝に対するイベント処理を行なうコマンド群

3. グラフに関する一般的な操作の機能．例えば，深さ優先探索，幅優先探索の機能．

Framework for Constructing Database Browser of Graph Structure
Akiko Katoand Kin'ichi Mitsuit
†IBM Research, Tokyo Research Laboratory

2.1 MVC のフレームワーク

MVC のフレームワークのうち，Model のクラス群はデータベースオブジェクト（「エレメント」と呼ぶことにする）とデータベースオブジェクト間の関係（同「リレーション」）を encapsulate するためのクラス群である．実際の表示画面上には，Model のクラス群のオブジェクトに対応する View クラス群のオブジェクトが表示される．ユーザーのインプットや Model と View のインタラクションを扱う Controller のクラスは，Smalltalk の MVC フレームワーク [1] では存在するが，我々の GvLibrary では存在しない．Controller の機能は Model クラスや View クラスのメンバー関数として実現され，これらメンバー関数は View オブジェクトのメニューやダイアログを通して呼び出される．

また通常の MVC フレームワークと同様に，Model オブジェクトの状態・性質の変化が起きた時に，対応する View オブジェクトにそれを伝達する機能を提供する．

ここで，我々の GvLibrary ではグラフ中の頂点・枝のみ Model/View に対応させていることに注意されたい．考えているアプリケーションによっては，データベース中のグラフの部分グラフを Model/View に対応させる必要が生じる場合がある．その部分グラフをいくつも表示するアプリケーションがそれに当たる．

以上の MVC フレームワークをもつ GvLibrary を使う場合，アプリケーションのデータベースに固有な部分は，GvLibrary で提供されている Model クラス群の適当なものから継承した Model クラスを定義することで対応できる．したがってアプリケーションを作る場合には Model クラスのみユーザーは定義すればよい．

2.2 View のグラフの表示・イベント処理に関する機能

View 側のグラフ全体の表示位置の移動，zooming，レイアウト戦略の切替え，overview ウィンドウの表示，View 側グラフの頂点や枝の色の変更，線の太さや種類の切替えなど，グラフ表示上の機能と，頂点・枝に対するイベント処理を行なう機能である．実際に頂点や枝の表示の変更がなされるのは，上記の MVC のフレームワークに従い，対応する Model の状態・性質が変更されたときである．また，グラフブラウザの基本操作の一

つである，グラフのインクリメンタルな展開や隠蔽を行なう機能も提供する．

2.3 一般的なグラフ操作の機能

データベースはグラフ構造を持つものを対象としているので，探索など一般的なグラフ操作の機能を提供する．たとえば，データベース中のあるエレメントからある指定されたリレーションで到達できる推移閉包を求めることを，ユーザーがそのエレメントとリレーションを指定すればできるような機能である．具体的には，GraphIterator クラスを提供し，ユーザーは始点となるエレメントに対応するオブジェクトと，リレーションに対応するエレメントのメンバー関数を指定する．

3 具体的な GvLibrary の階層構造

GvLibrary の具体的なクラスは，シェルウィンドウ，グラフを表示するエリア及び一般的なグラフ操作を扱う3つのクラスのほか，MVC フレームワークの機能を提供するクラス群からなる．MVC フレームワークの機能を提供するクラス群の階層を図 1に示す．アプリケーションを作る際にユーザーが定義しなければならないのは図 1の中で円で示されたクラスである．エレメントとリレーションの扱いの違いは，データベース中のエレメントに相当するクラスは必ず定義するが，リレーションについては定義してもしなくても良いと考えられることに対応する．リレーションに相当するクラスを定義しなければならないのは，リレーション自体に状態・性質がある場合で [2]，その変更にともなって表示を変えなければならない場合である．この場合の Model/View の対応は図 1中に示してある．リレーションに相当するクラスを定義しなくてもよいのは，リレーション自体には状態・性質がないと考えられる場合である．アプリケーションはデータベース中に存在するリレーションを直接参照するか，エレメントを通じてアクセスすることになる．，リレーションはエレメントに対応するクラスのメンバー関数として表現されるのみである．この場合は，リレーションに変更が起きた場合，GvLibrary の MVC の機能をリレーションに直接適用することができないので，すべてエレメントを通じて View の変更をおこなうことになる．その一方，ユーザーが不必要に多くのクラスを定義しなくてすむので，アプリケーションの設計が容易になり，アプリケーションは不必要に多くのオブジェクトを生成しない．この場合，Model/View の対応は Element/Node については図 1と同じだが，Arc に対応する Model クラスはない．

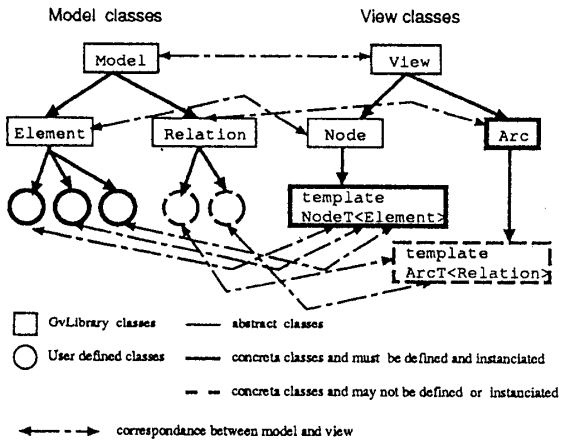

図 1: GvLibrary の Model/View のクラスの階層構造

図 1では各ユーザー定義の Model クラスに対応する View のクラスを C++のテンプレートクラスをインスタンシエートすることになっている．これにより，アプリケーションをつくるユーザーが，Model クラスのみを定義すれば良く，Model クラスに対応する View クラスを新たに定義する必要をなくす．テンプレートとするのは，Model クラスごとのメニューや，Model の変更のあった時の View の表示の変更のされ方（色・形など）が Model クラスのメンバー関数として表現されるので，それにアクセスするためである．但し，View クラスを定義することを厭わないユーザーは，Node・Arc クラスから継承してそれを定義することができる．

4 まとめ

以上述べてきたように，GvLibrary は，データベースをブラウズするアプリケーションを書くユーザーが，データベースに固有の部分の Model クラスのみ書けば，それを完成できるようになっている．但し，このためには，本来 View クラスの持つべき機能（Model クラスに固有のポップアップメニューの生成，View の表示の変更のされ方など）を一部 Model クラスのメンバー関数として書くことになる．そこで，View クラスに関しては GvLibrary の提供するテンプレートクラスを使うか，継承クラスを定義するかはユーザーにまかされている．

参考文献

[1] Kransner, G. E., and S. T. Pope: A cookbook for using the Model-View-Cotntroler user interface paradigm in Smalltalk-80, JOOP August/September 1988, pp.26–49.

[2] Rumbaugh, J., M. Blaha, W. Premerlani, F. Eddy, and W. lorensen: Object-Oriented Modeling and Design, Prentice Hall, 1991.