オブジェクト指向モデルに基づくバージョン管理機構のモデル化

4H−1

宮脇忠光*　干旭**　北川博之**　大保信夫**　藤原譲**
(*筑波大学理工学研究科　**筑波大学電子・情報工学系)

1. はじめに

近年、CADシステムやソフトウェア開発などにおける計算機利用の高度化に伴い、より効率的な設計データ管理の必要性が認識されてきている。設計データ管理においてはバージョン管理が重要である。従来の研究にはバージョンの履歴や構成関係の管理などを行なったものがある[1-4]。また論文[5]のようにバージョンに数種類の状態を導入しバージョンの設計段階によって状態を変化させられるモデルについて述べたものもある。本研究はバージョンの履歴、構成関係、状態とその変化などの情報の統合的管理を目的としている。本稿ではバージョンの履歴と状態変化の管理に着目し、設計対象の要求に応じた状態変化を管理できるバージョン管理機構のモデル化をオブジェクト指向モデルに基づいて行なった。

2. モデリング

以下に述べるクラス間の関係の概略を図1に示す。

2.1 バージョンとバージョンセット

互いに導出関係を持つバージョンの集合をバージョンセットと呼ぶ。以下にバージョンとバージョンセットを管理するためのクラスを示す。

●Versionクラス

個々のバージョンを記述する。バージョンの導出・削除・導出関係の問い合わせなどのユーザーインターフェースとなるオペレーションを持つ。ユーザー定義クラス（図1のProgramクラス）をバージョン管理する時にはこのクラスとユーザー定義クラスのサブクラス（図1のProgram_Vクラス）が定義される。よってVersionクラスから継承されるオペレーションを用いることによってクラスProgram_Vのオブジェクトはバージョン管理される。

●Vset_Mgrクラス

バージョンセットを管理する。このクラスのオブジェクトはVersionクラスオブジェクトと1対多の参照関係がある。このクラスは第一バージョンを生成するオペレーションを持つ。ユーザー定義クラス（図1のProgramクラス）をバージョン管理する時にはそのバージョンセットを管理するためにVset_Mgrクラスのサブクラス（図1のProgram_Vset_Mgrクラス）が定義される。

2.2 バージョンヒストリー

バージョンの導出関係とその時間情報をバージョンヒストリーと呼ぶ。バージョンヒストリーにおける導出関係は一般に木構造を成す。以下にバージョンヒストリーを管理するためのクラスを示す。

●Hist_Mgrクラス

バージョンヒストリー全般の管理をする。またバージョンヒストリー中の最新のバージョン、次のバージョンナンバーなどの管理をする。Vset_Mgrクラスのオブジェクトが生成されるとそのオブジェクトが管理するバージョンセットのバージョンヒストリーを管理するためにHist_Mgrクラスのオブジェクトが生成される。Vset_MgrクラスオブジェクトとHist_Mgrクラスオブジェクトは1対1の参照関係により対応付けられる。

●Hist_Nodeクラス

バージョンヒストリーの木構造の各ノードに対応する。各バージョンの生成時刻、削除時刻、バージョンナンバー、導出関係などの情報を管理する。バージョンヒストリーの木構造はHist_Nodeクラスオブジェクト間の参照関係として表現する。

2.3 バージョンの状態変化

一般にバージョンには設計段階に応じて様々な状態が考えられる。例えば論文[5]ではtransient,working,releasedの3種類が導入されている。transientは設計の初期段階であるため更新・削除が可能な状態である。workingは中間的な段階で削除は可能であるがグループで共有されるため更新できない状態である。releasedは最終段階であるため更新・削除が不可能な状態である。よってtransientからworkingへというような状態変化によってバージョンに対する更新・削除操作の適用が変化する。このような様々な設計段階に応じたバージョン管理を行なうために本モデルでは明示的に状態を表わすクラスとオペレーションが適用可能な状態と状態変化先を表わすクラスを定義した。これらによってバージョンは状態を持つことができ、オペレーションによって適用できる状態は決まる。オペレーションが適用できる状態と状態変化を管理するために図2のようなオペレーション管理表を用いる。図2は例えばVersionクラスに定義されているオペレーションDERIVEは3状態全てで適用可能であり、transientの場合にworkingへの状態変化を起こすことを示している。Versionクラスのオペレーションとユーザー定義クラスのオペレーションは全てこの表に登録されている。このオペレーション管理表を反映するために以下のクラスを定義した。

●Stateクラス

バージョンの取り得る状態を表わす。バージョンの現在の状態はバージョンが参照しているこのクラスのオブジェクトによって決まる。

●State_Logクラス

バージョンの状態変化の履歴を記録するクラス。状態とその変化時刻を記録する。次の状態への参照と変化時刻を保持する。

●Op_Exec_Ctlクラス

オペレーションが適用可能な状態と状態変化先を管理するオブジェクトのクラス。オペレーションとOp_Exec_Ctlクラスオブジェクトは1対1に対応している。オペレーションがある状態で適応可能かどうかを規定するPermissionクラスオブジェクトを参照する。

●Permissionクラス

オペレーションが適用可能な状態と、状態変化先を記述するオブジェクトのクラス。Stateクラスオブジェクトを参照することによって適用可能な状態と変化先を表わす。

3. 状態変化を伴うバージョン管理の例

バージョンには2.3で挙げた3種類の状態があるとする。例としてクラスProgramをバージョン管理する。このクラスの属性とオペレーションは図3のようになっている。システム定義クラスVersionとユーザー定義クラスProgramのオペレーション管理表は図2とする。

3.1 バージョン管理クラスの定義

ユーザー定義クラスProgramのバージョン管理のために以下の

Object-Oriented Version Management Modeling
Tadamitsh Miyawaki,Xu Yu,Hiroyuki Kitagawa,Nobuo Ohbo,Yuzuru Huziwara
University of Tsukuba

ような操作が行なわれる。

（1）Vset_Mgrのサブクラスとして Program_Vset_Mgrがシステムによって定義される。

（2）クラスProgramとクラスVersionのサブクラスProgram_Vがシステムによって定義される。

3.2 状態と状態管理表の登録

状態変化を伴うバージョン管理を行なうために必要な状態の登録をする。そしてシステム定義クラスVersionとユーザー定義クラスProgramのオペレーション管理表を登録する。

＜1＞状態の登録

（1）クラスStateに3状態を登録する。

（2）初期状態（transient）をクラスStateに登録する。新しく生成されるバージョンは必ずこの状態から始まる。

＜2＞オペレーション管理表の登録

（1）登録されるオペレーションに対してOp_Exec_Ctlオブジェクトを生成する。このオブジェクトが登録されたオペレーションに関する状態管理を行なう。

（2）（1）で登録されたオペレーションが適用可能な状態と状態変化先をPermissionクラスに登録する。

3.3 バージョンの生成と操作例

ここではバージョン管理クラスと登録したオペレーション管理表を用いてバージョンの生成と状態によって適用可能性が変化するオペレーションの例を挙げる。

（1）Program_Vset_MgrオブジェクトProgram_VMを生成する。これに連動して対応するHist_Mgrオブジェクトが生成される。

（2）オブジェクトProgram_VMに対してオペレーション CREATE_FIRST("Program_1")を用いることによって第1バージョンがクラスProgram_Vのオブジェクトとして生成される。オブジェクトProgram_1の状態は初期状態のtransientとなる。内部ではこれに連動して対応するHist_Nodeオブジェクトが生成され最初のバージョンナンバーがセットされる。さらにState_Logオブジェクトが生成されて現在の状態transientが登録される。

（3）オブジェクトProgram_1に引き数の登録をするためにオペレーションset_arg_list(args1)を用いる（図3）。このオペレーションが現在の状態（transient）で適用可能かどうかをクラスPermissionを検索して調べる。この場合オペレーション管理表（図2）に登録されているので現在は適用可能な状態である。よってオペレーションが実行される。

（4）初期の設計段階が終了した後オブジェクトProgram_1にオペレーションPROMOTE()を用いる。（3）と同様に適用可能である。そして状態変化を伴うので変化先である状態workingへとProgram_1の状態が変化する。内部では新たにState_Logオブジェクトが生成され次の状態としてworkingが登録される。

（5）再びオブジェクトProgram_1にオペレーションset_arg_list(args2)を用いる。（3）と同様に検索した結果現在の状態（working）では適用可能ではない。

この例は設計段階が進んだことにより、更新操作ができなくなったことを表わしている。このように状態変化を起こすことによって各設計段階における操作を制御することが可能となった。

4. おわりに

本研究では、履歴と状態変化の管理に着目したバージョン管理機構のモデル化をオブジェクト指向モデルに基づいて行なった。現在オブジェクト指向データベースONTOS上で実験システムを構築中である。今後さらにモデルの詳細検討と状態変化を伴うユーザー定義オペレーションへの対応などを目的とした検討を行なっていく予定である。

参考文献

[1]Randy H. Katz,"Toward a Unified Framework for Version Modeling in Engineering Databases",ACM Computing Surveys,Vol.22,No.4,December 1990.

[2]Rafi Ahmed,Shamkant B. Navathe,"Version Management of Composite Objects in CAD Databases",ACM,1991.

[3]Anders Bjornerstedt,Chirister Hunlten,"Version Control in an Object-Oriented Architecture".

[4]Peter Klahold,Guter Schlageter,and Wolfgang Wilkes,"A General Model for Version Management in Databases",Proc. of the Twelfth International Conference on VLDB, Kyoto,August,1986.

[5]Hong-Tai Chou,Won Kim,"Version and Change Notification in an Object-Oriented Database System",25th ACM/IEEE Design Automation Conference,1988

クラス	オペレーション	適用可能状態	変化先
Version	DELETE	transient	
		working	
Version	DERIVE	transient	working
		working	
		released	
Version	PROMOTE	transient	working
		working	released
・	・	・	・
・	・	・	・
Program	set_designer	transient	
Program	get_designer	transient	
		working	
		released	
Program	set_arg_list	transient	
・	・	・	・
・	・	・	・

図2 オペレーション管理表

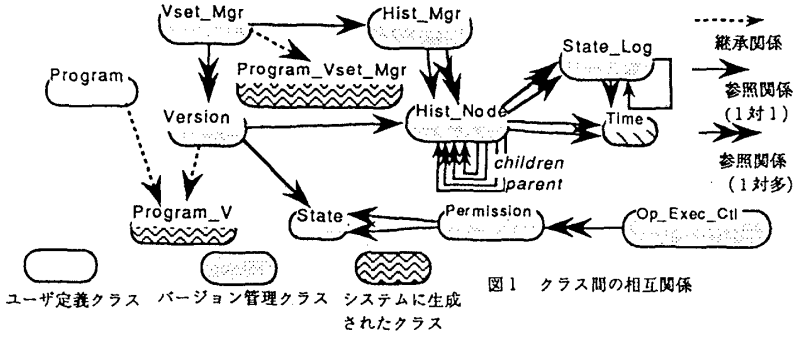

図1 クラス間の相互関係

```
class Program : Object
{ char* designer; /* 設計者名 */
  char* arg_spec; /* 引き数 */
        ・
        ・
        ・
public :
 void  set_designer(char* designer);

 char *get_designer();

 void  set_arg_list(char *arg_spec);
       /* 引き数の登録 */
        ・
        ・
        ・
};
```

図3 クラスProgram