データベースシステム 92－4
（1993. 3. 22）

アノテーションの概念を用いた応用適応型バージョン管理機構

宮脇 忠光*，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

エンジニアリング分野等におけるデータベースシステムの利用拡大に伴い，バージョン管理の重要性が認識されている．しかし，バージョン管理に対する要求は管理対象となるオブジェクトの種類や利用環境によって様々であるため，それぞれの応用目的に適したバージョン管理方式を提供する必要がある．本研究は，その一つのアプローチとしてセットメンバーアノテーションの概念を提案し，それをベースとした各種応用目的に適応可能なバージョン管理機構の実現について述べる．本論では，セットメンバーアノエーションの概念をオブジェクト指向データモデルに導入し，具体的エンジニアリングデータベースの例として，ソフトウェア開発支援データベースを取り上げ，そのバージョン管理機構の実現を示す．

An Application Adaptable Version Management Scheme Based on Set-Member Annotations

Tadamitsu Miyawaki*, Hiroyuki Kitagawa**, Nobuo Ohbo**, and Yuzuru Fujiwara**
*Master's Degree Program in Sciences and Engineering
**Institute of Information Sciences and Electronics
University of Tsukuba

In support of engineering database applications such as CAD, a version management is a indispensable feature required for database systems. However, application requirements for version management vary depending on target object types and design environments. Therefor, a version management scheme flexibly adaptable to application requirements has to be provided. This paper proposes an application adaptable version management scheme based on a notion of the set-member annotation. We show an object-oriented data model incorporating the set-member annotation and demonstrate development of a software development database system, which provides version management functions meeting application requirements.

1. はじめに

近年，エンジニアリング分野等のデータベースアプリケーションの多様化に伴い，データベースにおけるバージョン管理の重要性が認識されている[1,2].

バージョン管理の必要なデータベースアプリケーションの一例としてソフトウェアの開発が挙げられる．開発作業は通常既存のソフトウェアオブジェクトに変更を加えることによって進められる．開発を進めた結果，ある一定の水準に達したオブジェクトはリリースされる．またソフトウェア開発では外部から導入したソフトウェアシステムやモジュールを部品として利用する場合もある．

このようなソフトウェアの開発を支援するためには管理対象に応じたきめ細かなバージョン管理機構の実現が要求される．例えば，開発中のオブジェクトのバージョン管理においては，「どのバージョンに変更を加えて導出されたバージョンであるか」（導出関係）というバージョン間の関係や生成時刻等の情報を管理すると共に，既存のバージョンを基に新たなバージョンを導出する手続き等が必要である．また，リリースされたオブジェクトのバージョン管理においては，「どのバージョンの機能を継承したバージョンであるか」（機能的継承関係）というバージョン間の関係やリリース時刻，リリース管理者などのリリースされたバージョンに特有の情報を管理すると共に，ある一定の水準に達したかどうかをチェックした後にリリースバージョンとして登録する手続き等が必要である．さらに，導入したソフトウェアシステムやモジュールにも各種のバージョンが存在するためその管理が必要である．しかし，それらは直接的に開発の対象ではないため，導出関係・機能的継承関係のようなバージョン間の関係を管理する必要は通常無く，導入時刻や導入担当者等のような導入オブジェクトのバージョンに特有の情報を管理する必要があり，また導入ソフトウェアやモジュールをバージョンとして登録する手続きが必要である．

リリースされたオブジェクトをさらに改良するために新たなバージョンを導出することも一般に行なわれるため，リリースされたオブジェクトに対しては上で述べたようなリリースされたバージョンの管理のための情報のみならず，開発途中のバージョンと同様に導出関係等の情報をも管理する必要がある．

さらに，一般に設計環境では，あらかじめバージョンとして生成されずにテスト的に生成されたオブジェクトが開発作業に利用できることが判明し，後からそれをバージョン管理の対象にする必要が発生するというケースもまれではない．

以上のような要求は，ソフトウェア開発に限ったものではなく，一般の設計データにおけるバージョン管理にも頻繁に生じることである．従って，データベースシステムの提供するバージョン管理機構は，以下のような条件を満足するものであることが望まれる．

1）管理対象に応じたバージョン管理に必要な情報の登録や手続きの付加が可能であること．
2）同一のオブジェクトを複数のバージョン管理方式で管理できること．
3）オブジェクトを生成後にバージョン管理の対象にできること．

これまでにも，バージョン管理に関連して関係データモデル，関数データモデル，オブジェクト指向データモデル等を基に様々な研究が行なわれてきた．また，DBMSの中にはバージョン管理機構を提供しているものもあるが，それらのほとんどは全てのオブジェクトに画一的なバージョン管理機構を提供しているため，応用目的に適応したバージョン管理を行なうことが難しい[3-10].

これに対し，近年，応用目的に適応したバージョン管理機構の実現方式を目指した研究も幾つかある[11-17]．オブジェクト指向データモデルに基づくいくつかの研究[13-16]では，バージョン管理の対象となるオブジェクトのタイプに応用目的に応じたバージョン管理のためのデータ要素や手続きを定義し，バージョンを生成・管理するというアプローチをとることにより，上記1）の要求事項への対応を図っている．

しかし，これらの方式では通常以下のような問題が生じる．

（1）データベーススキーマの設計時に，対象となるオブジェクトの属性や手続きの他にバージョン管理の方式まで考慮して各タイプを設計しなければならないため，どのタイプのオブジェクトをどのような方式でバージョン管理するかをデータベース設計時に全て決定しておかなければならない．また，これにより，一般にデータベーススキーマが複雑化する．

（2）バージョンの生成時にバージョン管理方式

が決定されるため，開発作業の進展に伴い複数のバージョン管理方式で同一オブジェクトを管理するのが難しい．従って，上記2）の要求事項への対応が困難である．

（3）オブジェクトの生成時にそのオブジェクトをバージョン管理するかどうか，またその管理方式が決定されてしまうため，上記3）の要求事項への対応が困難である．

本論文では，これらの問題に対するアプローチとしてセットメンバーアノテーションという概念を提案し，それを利用することにより上記の1）～3）の要求に対応したバージョン管理機構が実現可能であることを示す．

以下，2章で本研究におけるデータモデルを定義する．3章では本研究で提案するセットメンバーアノテーションの概念について説明する．4章ではセットメンバーアノテーションに基づくバージョン管理機構の実現方法について述べる．5章ではソフトウェア開発データベースを例として1）～3）の要求に対応したバージョン管理機構の実現とその操作例について述べる．6章ではセットメンバーアノテーションのバージョン管理における有効性を検討する．7節ではまとめを行なう．

2. データモデル

本研究ではベースとなるデータモデルとして，C++ベースの標準的OODBMSにおけるデータモデルを想定している．本論文での議論を具体的なものにするために商用OODBMS ONTOS[19]のデータモデルをベースとしたデータモデルを以下に示すが，本論文での議論は一部の修正により他の標準的なOODBMSに適用可能であると考えられる．

定義1　オブジェクトとクラス

実世界の実体（entity）をオブジェクトとして表わす．オブジェクトには，プログラムの終了時に消える揮発オブジェクトと消えない永続オブジェクトがある．データベースは永続オブジェクトの集まりによって構成される．永続オブジェクトはデータベース中でユニークな識別子を持つ．

本モデルでは，オブジェクトの型としてC++の基本データタイプ（char, int, short, long, enum等）を用いることができる．また，C++と同様にユーザー定義タイプを定義できる．このタイプのことをクラスと呼ぶ．クラスはそのクラスを表現するデータメンバー，そのクラスに対する操作を実現するメンバー関数，そして少なくとも一つのコンストラクタからなる．メンバー関数とは異なり，どのクラスにも属さない関数（フリー関数）を定義可能である．

定義2　継承

クラスを定義する時に類似した部分を持つクラスがある場合，そのクラスを拡張して新たなクラスを導出することができる．基になったクラスを基底クラス，導出されたクラスを導出クラスと呼ぶ．基底クラスに定義されたデータメンバーとメンバー関数は導出クラスに全て継承される．導出クラスにはそのクラス独自のコンストラクタ，データメンバー，メンバー関数を定義することができる．

導出クラスには基底クラスを複数指定することができ，多重継承と呼ぶ．この場合，全ての基底クラスのデータメンバーとメンバー関数が導出クラスに継承される．基底クラス中に同じ名前のデータメンバー，メンバー関数が定義されている場合，どの基底クラスから継承されたものを表わすのか曖昧になる．この曖昧さを解決するために，本モデルでは<基底クラス>::<データメンバーまたはメンバー関数>の形式で基底クラスを明示的に指定する．

定義3　永続クラス

プログラムが終了しても消えない永続オブジェクトは，永続クラスのオブジェクトとして生成される．永続クラスはObjectクラスの直接的または間接的な導出クラスでなければならない．また永続クラス自身は，Typeクラスのオブジェクトとしてデータベースに格納される．Objectクラスには，以下のメンバー関数が定義されている．

- Type* getDirectType()
  永続オブジェクトが属するクラスオブジェクトを返す．
- void putObject()
  永続オブジェクトをデータベースに格納する．
- void deleteObject()
  永続オブジェクトをデータベースから削除する．

永続オブジェクトの生成はC++における通常のオブジェクトの生成方式に従う．但し，データベースがオープンされている間にしか生成することができない．

定義4　Setクラス

重複の無いオブジェクトの集まりをオブジェクトとしてモデル化するためにSetクラスを導入する．SetクラスはObjectクラスの導出クラスであり，永続クラスである．Setクラスのオブジェクトは，生成時にメンバーとなるオブジェクトのクラスを指定される．Setクラスは以下のメンバー関数を提供する．

・OC_Boolean isMember(Object* theMember)
　指定したオブジェクトがメンバーならばTRUEを返し，メンバーでなければFALSEを返す．
・Type* memberSpec()
　メンバーとして含むことができるオブジェクトのクラスオブジェクトを返す．
・unsigned long Cardinality()
　現在メンバーとして含まれているオブジェクトの数を返す．
・void Insert(Object* theMember)
　オブジェクトをメンバーとして登録する．
・void Remove(Object* theMember)
　オブジェクトをメンバーから削除する．

定義5　Iteratorクラス

Setクラスのオブジェクトに含まれるメンバーオブジェクトにアクセスする機能を提供するクラスである．Iteratorクラスのオブジェクトは生成時にSetクラスのオブジェクトを指定され，その指定されたオブジェクトのメンバーオブジェクトを一つづつ重複無く取り出すことができる．IteratorクラスはObjectクラスの導出クラスではない．Iteratorクラスは以下のメンバー関数とオペレーターを提供する．

・Reset()
　Iteratorクラスのオブジェクトの状態をリセットする．
・moreDate()
　メンバーがあればTRUEを返し，無ければFALSEを返す．
・()
　メンバーオブジェクトを一つ取り出す．

3.　セットメンバーアノテーション

本章では，本研究で提案するセットメンバーアノテーションの概念について説明し，2章で述べたオブジェクト指向データモデルへの導入について述べる．

3.1　アノテーションの概念の導入

アノテーションは，主にアクセス指向プログラミングの分野で導入されている概念であり，オブジェクトのデータ要素に対して用いられている[18]．アノテーションを用いることにより，データ要素の通常の値に加えて，データ要素のデフォルト値，過去の値の履歴等のような付加的な情報をデータ要素に付け加えることができる．

本研究では，集合中の各メンバーオブジェクトに対してアノテーションの概念を適用するセットメンバーアノテーションという概念を提案する．

セットメンバーアノテーションの概念では，集合のメンバーとすることによって，オブジェクトに動的にあるデータメンバーとメンバー関数を持つアノテーションオブジェクトを付け加えることができる（図1）．このような機能を持つ集合をアノテーション付き集合と呼ぶ．メンバーオブジェクトとアノテーションオブジェクトの対応付けはアノテーション付き集合によりなされる．あるオブジェクトが複数のアノテーション付き集合に属する場合には，それぞれ別のアノテーションオブジェクトが付け加えられる．あるメンバーオブジェクトがアノテーション付き集合のメンバーでなくなった場合には，自動的にアノテーションオブジェクトとの対応は解消される．

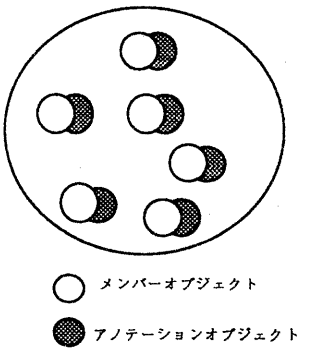

図1　セットメンバーアノテーション

3.2　データモデルへの導入

本節では，2章で定義したオブジェクト指向データモデルへのセットメンバーアノテーションの導入について述べる．

セットメンバーアノテーションの概念は，Setクラスの導出クラスでありアノテーション付き集合オブジェクトを表わすクラスであるASetクラス

と，アノテーションオブジェクトの属するクラスであるアノテーションクラスによって実現される．ASetクラスは，Setクラスと同様にオブジェクトをメンバーとして登録された際，アノテーションオブジェクトの生成とメンバーオブジェクトとアノテーションオブジェクトの対応付けを行なう．以下にASetクラスが提供するコンストラクタとメンバー関数，そしてASetクラスに関連するフリー関数を示す．

（1）コンストラクタとメンバー関数

1. ASet(Type* theMemType, Type* theAnnType)
   第1引数にメンバーとなるオブジェクトのクラスを，第2引数にアノテーションクラスを指定する．
2. void Insert(Object* theObject)
   指定したオブジェクトをメンバーとして登録する．この時対応するアノテーションオブジェクトが生成される．
3. void Remove(Object* theObject)
   指定したオブジェクトをメンバー中から削除する．この時対応するアノテーションオブジェクトが削除される．
4. Object* Annotation(Object* theMember)
   指定したメンバーオブジェクトに対応するアノテーションオブジェクトを返す．無ければNULLを返す．
5. Object* Member(Object* theAnnotation)
   指定したアノテーションオブジェクトと対応するメンバーオブジェクトを返す．無ければNULLを返す．

（2）関連するフリー関数

1. Object* OC_getASetfromMember(Object* theMember)
   指定したオブジェクトがメンバーとして含まれるASetクラスのオブジェクトを返す．複数の場合はASetクラスのオブジェクトをメンバーとするSetクラスのオブジェクトを返す．無ければNULLを返す．
2. Object* OC_getASetfromAnnotation(Object* theAnnotation)
   指定したアノテーションオブジェクトと対応するメンバーオブジェクトを含むASetクラスのオブジェクトを返す．無ければNULLを返す．

3.3 セットメンバーアノテーションの実現

セットメンバーアノテーションの機構をOODBMS ONTOS DB 2.2[19]上で実現した．

ASetクラスはメンバーオブジェクトとアノテーションオブジェクトの対応付け機能を提供する．この機能をONTOSが提供するNameとDirectoryの機能を利用して実現した．

ONTOSでは永続オブジェクトに名前を付けることができ，その名前を任意のDirectoryに登録することができる．この名前によってオブジェクトを直接参照することができる．Directoryが異なれば，異なるオブジェクトが同一名を持ったり，同一オブジェクトが複数の名前を持つことができる．

ONTOSが提供するこの機能を用いて，ASetオブジェクトは，メンバーオブジェクトとアノテーションオブジェクトのそれぞれに対応付け可能な名前を付け，それぞれ異なるDirectoryに名前を登録することにより対応を管理している．

4. セットメンバーアノテーションに基づくバージョン管理機構

本章では，セットメンバーアノテーションに基づくバージョン管理機構の実現について述べる．

4.1 実現方針

以下の2種類のクラスを定義することによりセットメンバーアノテーションの概念に基づくバージョン管理機構を実現することが可能である．

（1）ASetクラスの導出クラス

バージョン管理においては，ある設計対象に対応して作成された一群のバージョンの集まりをバージョンセットとして管理する必要がある．本クラスはバージョンセットを表現するオブジェクトの属するクラスである．本クラスには，バージョン操作用のメンバー関数を定義する．バージョンセットのメンバーとなる各バージョンにバージョン管理用の情報をアノテーションオブジェクトとして付加できるよう，本クラスはASetクラスの導出クラスとする．

（2）アノテーションクラス

バージョンセットのメンバーオブジェクトとして管理する各バージョンに付加するアノテーションオブジェクトを規定するクラスとして本クラス

を定義する．本クラスには，バージョン管理用のデータメンバーやメンバー関数を定義する．

4.2 実現例

セットメンバーアノテーションの概念に基づくバージョン管理機構を4.1節の実現方針に従って実現する．例として，1章で述べた開発途中のオブジェクトのバージョン管理を想定した機構を考える．

開発途中のオブジェクトのバージョン管理においては，互いに導出関係のあるバージョンの集まりを開発バージョンセットとして管理し，その中のバージョンを識別するために各バージョンにバージョンナンバーを付ける必要がある．また，指定した時刻における最新のバージョン等を調べるために，各バージョンの登録時刻を管理すると共に，既存のバージョンを基に新たなバージョンを導出する手続きとテスト的に生成されたオブジェクトをバージョン管理の対象にする手続きが必要である．

以上の様なバージョン管理機構を実現するためには，図2のDesignVerSetクラスと図3のDesignVersionクラスを定義する．

DesignVerSetクラスは，開発バージョンセットを表現するクラスであり，このクラスに開発途中のオブジェクトに特有の手続きをメンバー関数として定義する．これらは，ASetクラスから継承されるメンバー関数とDesignVersionクラスのメンバー関数を組み合わせることによって実現される．

登録用手続きとしては，指定したオブジェクトを最初のバージョンとして登録するFirstVersionと，新たなバージョンを導出するDeriveを定義する．

バージョンナンバーを参照するVersionNo，バージョンナンバーからバージョンを参照するVersion，時刻に関する参照手続きとして，登録時刻を参照するRegistTime，指定した時刻における最新バージョンを参照するMostRecentが定義される．導出関係の参照手続きとして親バージョンを参照するParent，指定した時刻における子バージョンを参照するChildを定義する．さらに，時間的に直前・直後に登録されたバージョンに関する参照手続きとしてPredとSuccを定義する．また，Parent, Child, Pred, Succのそれぞれに対して推移的閉包をとるメンバー関数（_Parent, _Child等）を定義する．

```
class DesignVerSet : ASet
{
  Reference mostRecent;
public :
  DesignVerSet(Type* theAnnType);
  void FirstVersion(Object* theVersion);
  Object* Derive(Object* theBaseVersion);
  int VersionNo(Object* theVersion);
  Object* Version(int theVersionNo);
  Time* RegistTime(Object* theVersion);
  Object* MostRecent(Time* theTime);
  Object* Parent(Object* theVersion);
  Set* Child(Object* theVersion,
             Time* theTime);
  Object* Pred(Object* theVersion);
  Object* Succ(Object* theVersion,
               Time* theTime);
             ...
};
```

図2 DesignVerSetクラス

DesignVersionクラスは，開発途中の各バージョンに固有の付加情報を表わすためのアノテーションクラスである．このクラスには，バージョンナンバー，登録時刻，導出関係，時間的に直前・直後のバージョンの情報を記録・管理するためのデータメンバーとメンバー関数を定義する．

```
class DesignVersion : public Object
{
  int versionNo;
  Reference registTime;
  Reference parent;
  Reference child;
  Reference succ;
  Reference pred;
public:
  int GetVersionNo();
  void SetVersionNo(int theVersionNo);
                  ...
};
```

図3 DesignVersionクラス

5. 応用例

本章では，ソフトウェア開発データベースを例としてとりあげ，それを対象として1章の1）～3）の要求事項に対応したバージョン管理機構の実現とその操作例について述べる．

5.1 データベースの概要

本ソフトウェア開発データベースのデータベーススキーマの概要は図4に示すものである．図中には各種のクラスが存在するが，実線矢印は1対1参照関係，実線二重矢印は1対多参照関係，そして波線矢印は継承関係を示す．

開発対象システムは，複数のプログラムとデータファイルから構成され，マニュアルとライブラリを提供する．プログラムは複数のヘッダーとパ

ッケージから構成される．各ヘッダーはヘッダーコードを持ち，また内部的に他のヘッダーを参照することもある．パッケージは，パッケージコードを持ち，また複数のヘッダーを参照する．開発対象システムとその構成要素には全て仕様書があり，開発者が存在する．

また開発対象システムは，複数の導入システムを利用して開発される場合もある．導入システムは，それぞれ複数のヘッダー，ヘッダーコード，ライブラリ，マニュアルを提供する．開発対象システムのプログラムを構成するヘッダーとライブラリの中には導入システムによって提供されるものもある．（DSystemクラスとESystemクラスのように類似した部分を持つクラスには全て共通の基底クラスを実際には定義しているが，図が煩雑になるため，ここでは省略している．）

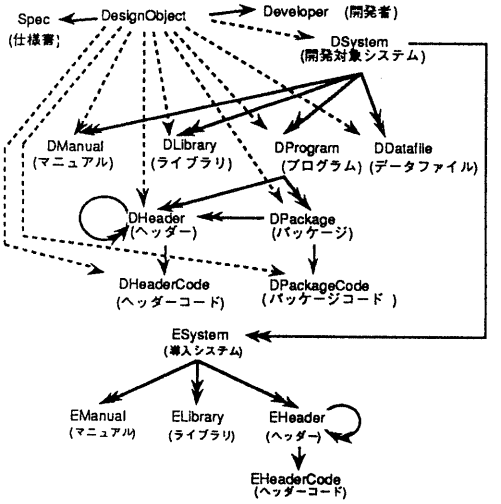

図4　ソフトウェア開発データベース

5.2　バージョン管理機構に対する要求

本ソフトウェア開発データベース中には，開発途中のオブジェクト，リリースオブジェクト，導入オブジェクトの3種類のオブジェクトが存在する．序論で述べたように，これらのオブジェクトは，それぞれ異なるバージョン管理方式で管理する必要がある．

開発途中のオブジェクトのバージョン管理については4.2節で述べた．

リリースオブジェクトのバージョン管理に関しては，リリースバージョンセット中で，オブジェクトを識別するために，リリースバージョンナンバーを管理する必要がある．また，リリースバージョン間での機能的継承関係を管理する必要がある．さらに，リリース時刻やリリース管理者というリリースオブジェクトに特有の情報を管理すると共に，各管理情報に対する参照手続きと開発途中のオブジェクトが一定の水準に達しかどうかをチェックした後にそれをリリースオブジェクトとして登録する手続きが必要である．

導入オブジェクトのバージョン管理に関しては，それらは直接的に開発の対象ではないため，導出関係・機能的継承関係のようなバージョン間の関係を管理する必要は無く，同じ系列の導入システムやモジュールの集まりを導入バージョンセットとして管理し，導入時刻，バージョンナンバー，そして導入担当者の情報を管理する必要がある．また，各管理情報に対する参照手続きの他に導入システムやモジュールをバージョンとして登録する手続きが必要である．

5.3　バージョン管理機構の実現

開発途中のオブジェクトに対するバージョン管理機構の実現は4.2節で示した．本節では，リリースオブジェクトと導入オブジェクトのバージョン管理機構を4.1節の実現方針に基づいて実現する．

リリースオブジェクトのバージョン管理機構

リリースオブジェクトのバージョン管理機構を実現するために，図5のReleaseVerSetクラスと図6のReleaseVersionクラスを定義する．

ReleaseVerSetクラスは，ASetクラスの導出クラスであり，リリースバージョンセットを表現するクラスである．このクラスには，指定した開発途中のオブジェクトがある水準に達しているかを調べた後にそれをリリースオブジェクトとしてリリースバージョンセットに登録するメンバー関数Registerが定義されている．この引数にどのリリースオブジェクトから機能を継承するのかを指定することができる．

参照手続きとしては，リリースバージョンナンバーを参照するReleaseNo，リリースバージョンナンバーからバージョンを参照するReleaseが定義されている．機能的継承関係の参照用メンバー関数として，親リリースバージョンを参照するParentRelease，指定した時刻における子リリースバージョンを参照するChildReleaseが定義されて

いる．機能的継承関係と時間的に直前・直後にリリースされたリリースバージョンを参照するメンバー関数に対して，それぞれ推移的閉包をとるメンバー関数が定義されている（_ParentRelease, _ChildRelease等）．

```
class ReleaseVerSet : public ASet
{
  Reference mostRecent;
public :
  ReleaseVerSet(Type* theAnnType);
  void Register(Object* theVersion,
                Object* theBaseVersion);
  int ReleaseNo(Object* theVersion);
  Object* Release(int theReleaseNo);
  Time* ReleaseTime(Object* theVersion);
  Object* MostRecent(Time* theTime);
  Object* ParentRelease(Object*
                        theVersion);
  Object* ChildRelease(Object* theVersion,
                       Time* theTime);
  Object* Pred(Object* theVersion);
  Object* Succ(Object* theVersion,
               Time* theTime);
                  …
};
```

図5　ReleaseVerSetクラス

ReleaseVersionクラスは，リリースされた各バージョンに固有の付加情報を表わすためのアノテーションクラスである．このクラスには，リリースバージョンナンバー，リリース時刻，機能的継承関係，リリース管理者の情報を記録・管理するためのデータメンバーとメンバー関数が定義されている．

```
class ReleaseVersion : public Object
{
  int relaeseNo;
  Reference releaseTime;
  Reference parentRelease;
  Reference childRelease;
  Reference pred;
  Reference succ;
  Reference releaseManager;
public:
  int GetReleaseNo();
  void SetReleaseNo(int theReleaseNo);
                  …
};
```

図6　ReleaseVersionクラス

導入オブジェクトのバージョン管理機構

導入オブジェクトのバージョン管理機構を実現するために，図7のExternalVerSetクラスと図8のExternalVersionクラスを定義する．

ExternalVerSetクラスは，導入バージョンセットを表現するクラスである．このクラスには登録用メンバー関数として，導入システムやモジュールをバージョンとして登録するRegisterが定義されている．また，バージョンナンバーに関する参照用メンバー関数（ExternalNo, External）を定義する．さらに，導入時刻に関する参照用メンバー関数（IntroTime, MostRecent, Pred, Succ）が定義されている．時間的に直前・直後に導入されたバージョンを参照するメンバー関数に対して，推移的閉包をとるメンバー関数が定義されている（_Pred, _Succ）．

```
class ExternalVerSet : public ASet
{
  Reference mostRecent;
public :
  ExternalVerSet(Type* theAnnType);
  void Register(Object* theVersion,
                int theVersionNo);
  int ExternalNo(Object* theVersion);
  Object* External(int theReleaseNo);
  Time* IntroTime(Object* theVersion);
  Object* MostRecent(Time* theTime);
  Object* Pred(Object* theVersion);
  Object* Succ(Object* theVersion,
               Time* theTime);
                  …
};
```

図7　ExternalVerSetクラス

ExternalVersionクラスは，導入オブジェクトのバージョンに固有の付加情報を表わすためのアノテーションクラスである．このクラスには，バージョンナンバー，導入時刻，導入担当者の情報を記録・管理するためのデータメンバーとメンバー関数が定義する．

```
class ExternalVersion : public Object
{
  int externalNo;
  Reference introTime;
  Reference pred;
  Reference succ;
  Reference chargePerson;
public:
  int GetExternalNo();
  void SetExternalNo(int theExternalNo);
                  …
};
```

図8　ExternalVersionクラス

5.4　バージョン操作例

本節では，DSystemクラスのオブジェクトをバージョン管理の対象として，本データベースにおけるバージョン操作例について示す．

1）第1バージョンの登録

DSystemクラスのオブジェクトを生成し，このオブジェクトをバージョン管理の対象にするために開発バージョンセットtheDVSetを生成する．メ

ンバー関数FirstVersionを用いて第１バージョンとして登録する（図９）．この時対応するアノテーションオブジェクトが生成され，FirstVersion内でDesignVersionクラスのメンバー関数が呼ばれてそのオブジェクトにバージョンナンバー，登録時刻，導出関係が登録される．

```
DSystem* theDSys1;
DesignVerSet* theDVSet;
…
//DSystemクラスのオブジェクトを生成
theDSys = new DSystem();
//開発バージョンセットを生成
theDVSet = new DesignVerSet(
                getDirectType(theDSys1));
…
//第１バージョンの登録
theDVSet->FirstVersion(theDSys1);
```

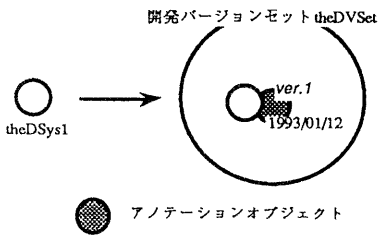

図９　第１バージョンの登録

２）バージョンの導出

theDSys1を基に新たなバージョンを導出する．この時対応するアノテーションオブジェクトが生成され，メンバー関数Derive内でDesignVersionクラスのメンバー関数が呼ばれてそのオブジェクトにバージョンナンバー，登録時刻，導出関係が登録される（図１０）．

```
DSystem* theDSys2;
…
//バージョンの導出
theDSys2 = theDVSet->Derive(theDSys1);
```

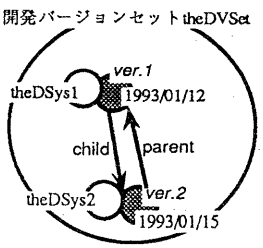

図１０　バージョンの導出

３）バージョンのリリース

図１１は２）の操作が繰り返された結果，リリースオブジェクトとして登録されているtheDSys3の機能を継承するリリースオブジェクトとして，theDSys6がリリースバージョンセットtheRVSetに登録された状態を示している．（図１１は時間情報と開発途中のオブジェクトのparentを省略している．）

```
ReleaseVerSet* theRVSet;
//リリースバージョンセットを生成
theRVSet = new ReleaseVerSet(
                getDirectTye(theDSys1));
…
//DSys6をリリース
theRVSet->Register(theDSys6, theDSys3);
```

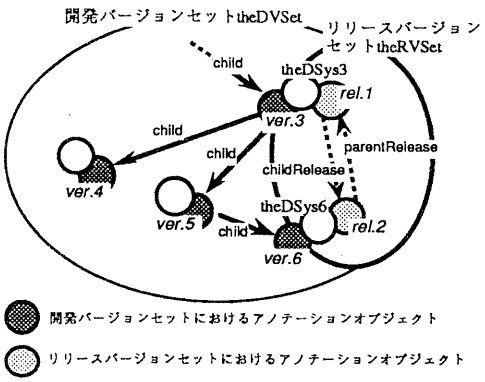

図１１　バージョンのリリース

７．評価

セットメンバーアノテーションの概念に基づくバージョン管理機構では，１章で述べた要求事項１）～３）に対応可能である．

要求事項１）に関しては，管理対象に応じたバージョン管理に必要な情報をアノテーションクラスのオブジェクトで管理し，Asetクラスの導出クラスのメンバー関数として必要な手続きを定義することによって対応可能である．

要求事項２）に関しては，５.４節で示したように開発バージョンセット中のオブジェクトをリリースバージョンセットに登録することが可能である．本方式では，このようにして複数のバージョン管理方式の下に管理可能である．

要求事項３）に関しては，５.４節で示したように既に存在しているオブジェクトを後からバージョンセット中に登録することが可能である．このように，オブジェクトを生成後にバージョン管理の対象にすることができる．

一方，上記の検討を通じて以下のような問題点も明らかになった．

（1）バージョンセットの特定

アノテーションオブジェクトによって付加されるバージョンとしての情報を知るためには，バージョンセットの特定が必ず必要である．

（2）バージョンの削除

メンバーオブジェクトをバージョンセットから削除すると対応するアノテーションオブジェクトも自動的に削除される．従って，例えばバージョンの実体は削除したいがその導出関係等の履歴情報は残したいというような要求に対応することが難しい．

8．おわりに

本論では，セットメンバーアノテーションの概念に基づくバージョン管理機構について述べた．さらに，ソフトウェア開発データベースを例としてバージョン管理機構の具体的実現について説明し，その操作例を示した．これらを通して，応用要求に対応したバージョン管理機構を実現する上でセットメンバーアノテーションの長所と問題点を明らかにした．また，標準的なオブジェクト指向データモデルの枠組みを大きく変更することなく，セットメンバーアノテーションの概念は導入可能であることも示した．

今後の研究課題としては，まず6章で述べた問題点の解決が挙げられる．また，応用目的に応じたASetクラスの導出クラスとアノテーションクラスの定義を容易とするため，様々な導出関係や時間情報の管理要求に対応するため，バージョン管理用の標準的なASetクラスやアノテーションクラスのライブラリについて検討する必要がある．

9．謝辞

本研究を進めるにあたり，終始ご指導，ご助言を下さった筑波大学データベース研究室の諸氏に深く感謝いたします．

参考文献

[1] M.Atkinson, et al, "The Object-Oriented Database System Manifesto", Deductive and Object-Oriented Databases.

[2] R.H.Katz, Information Management for Engineering Design.

[3] R.Snodgrass, "The Temporal Query Language TQuel", ACM TODS, 12(2).

[4] R.H.Katz, "Toward a Unified Framework for Version Modeling in Engineering Databases", ACM Computing Surveys, 22(4).

[5] R.Ahmed, and S.Navathe, "Version Management of Composite Objects in CAD Databases", ACM, 1991.

[6] H.T.Chou, and W.Kim, "Version and Change Notification in an Object-Oriented Database System", 25th ACM/IEEE Design Automation Conf., 1988.

[7] H.T.Chou, and W.Kim, "A Unifing Framework for Version Control in a CAD Environment", Proc. of the 12th VLDB, 1986.

[8] D.Beech, and B.Mahbod, "Generalized Version Control in an Object-Oriented Database", IEEE Proc. 4th Int'l. Conf. on Data Engineering, 1988.

[9] R.Agrawal, et al., "Object Versioning in Ode", IEEE Proc. 7th Int'l. Conf. on Data Engineering, 1991.

[10] 田中 肇 他，"履歴データ型を用いたバージョン管理方式の設計と実装"，情報処理学会第92回データベースシステム研究会.

[11] P.Klahold, et al., "A General Model for Version Management in Databases", Proc. of the 12th VLDB, 1986.

[12] K.R.Dittrich, and R.A.Lorie, "Version Support for Engineering Database Systems", IEEE Trans. on Software Engineering, 1988.

[13] A. Bjornerstedt, and C.Hulten, "Version Control in an Object-Oriented Architecture", Object-Oriented Cocepts, Databases, and Applications.

[14] 加藤 丈晴，"オブジェクト指向モデルに基づいた設計データのバージョン管理"，平成2年度筑波大学大学院修士課程理工学研究科修士論文.

[15] E. Sciore, "Multidimensional Versioning for Object-Oriented Databases", Second Int'l. Conf. DOOD, 1991.

[16] E. Sciore, "Using Annotation to Support Multiple Kinds of Versioning in an Object-Oriented Database System", ACM TODS, 16(3).

[17] G.T.J.Wuu, and U.Dayal, "A Uniform Model for Temporal Object-Oriented Databases", IEEE Proc.8th Int'l. Conf. on Data Engineering.

[18] M.Stefik, et al., "Integrating Access-Oriented Programming into a Multiparadigm Environment", IEEE Soft., 3(1).

[19] Ontos, Inc., ONTOS DB 2.2 Developer's Guide.