基盤技術としてのリポジトリ——UREP を中心として

Repository as Base Technology——Focusing on UREP

行　成　　教

要　約　リポジトリは，CALS アーキテクチャの基盤的役割を担い，CALS 実装において欠かせないものである．そこで，本稿では基盤技術としてのリポジトリに焦点を当てて議論する．まず，CALS アーキテクチャの現状と課題について述べ，リポジトリの動向について説明する．次に，リポジトリ一般論を展開した後，米国ユニシス社が開発したユニバーサル・リポジトリ（UREP）の機能および適用事例について説明する．最後に，今後の展望について述べる．

Abstract　The repository plays a central role in the CALS architecture and is indispensable to CALS implementation. This paper focuses on the repository as a base technology. First, it discusses current CALS architecture and problems, followed by an introduction to repository standardization activities and industry trends. Next, the general features of repository are described and compared with Universal Repository (UREP) developed by Unisys. A UREP case study is also included. This paper concludes with a look at the future direction of CALS mainly in connection with repositories.

1. は　じ　め　に

ソフトウェア，ハードウェアを含む構成情報等，情報資源を管理するためのデータをメタデータと呼ぶ．メタデータの種類には，表1に示すようなものがある．

表 1　メタデータの種類

定義情報	●設計情報，モデル情報，プログラム情報など ●データウェアハウスのためのメタデータ
ビジネス情報	●ビジネス・ルール，プロセス，ポリシなど
ランタイム情報	●システム構成情報，バックアップ情報など

これらの情報の特徴は，

1）　利用者のデータベースに比べて，データ量が比較的少量であること

2）　個々のデータ間に複雑な関連を持つこと

3）　ある頻度で動的に変更されること

である．このメタデータを管理・格納するためのデータ格納庫がリポジトリである．

メタデータ管理の必要性はかなり以前から提案され実装も試みられてきたが，実際に成功した例はあまり見当たらない．その原因として，

1）　投資対効果を見たとき，本格的なリポジトリを作成するより，データベース管理システム等を利用して，そのシステムに特化したデータを格納・管理する方が有効であると思われていたこと．

2）　規格としての標準化は進められていたが，リポジトリを支える基盤技術が未発

達で，実用に耐えられるプロダクトがなかったこと．

が考えられる．

前者については，近年コンピュータ環境におけるマルチベンダー化・分散化に伴い，その環境を管理するためのリポジトリの必要性が高まっている．特に，CALS 等文化の異なる企業，組織間でデータを交換する場合，どこにどのような情報がどんな形で存在するのか管理して利用者に公開する必要があり，リポジトリに対するニーズは高い．

後者については，オブジェクト指向技術の発達とともに，複雑な構造を持つメタ・データを管理する技術基盤が整い，それに即したプロダクトが市場に登場している．

このような状況を踏まえて，本稿では CALS で重要な位置を占めると思われるリポジトリの現状，問題点および課題を述べる．

2 章では，CALS におけるリポジトリの役割を説明し，現状の課題について述べる．3 章では，リポジトリをめぐる標準化動向および業界動向について簡単に触れ，4 章では，リポジトリ機能について説明する．5 章では，米国ユニシス社のリポジトリ製品ユニバーサル・リポジトリ（UREP）の機能について説明し，6 章で，UREP の適用事例を紹介する．7 章で CALS の問題点とそれを解決するための UREP の提供する機能について説明し，8 章で今後の展望について述べる．

2. CALS とリポジトリ――CALS での役割と課題

CALS は異なる企業間で合意された業務手順に従い，標準化された形式の電子データを交換・共有することによってビジネスを行う概念である．その中で，ソフトウェア CALS は，ソフトウェア部品，すなわちモジュールやコンポーネントを企業間で交換することを一つの目的とする．

近年ソフトウェアの大規模化に伴って，ソフトウェア開発に多数の人手を必要とし，ソフトウェアすべての部分を一つの会社で開発することは不可能になった．

一方，ソフトウェア技術において，オブジェクト指向の実用化とコンポーネントウェアの普及により，ソフトウェアの部品化・再利用が促進され，ソフトウェア部品を広く共有する考えが生まれてきた．また，標準部品を作成することで，ビジネスが成り立ち，専門家による部品開発が進められている．

このように，複数の企業がソフトウェア部品の発注・受注関係で結び付くことに対する要求の高まりと技術基盤の進展により，CALS が注目されるようになった．

そこで必要となるのが，部品の発注・受注において，どのような情報を要求書に盛り込むか，成果物をどのようにやり取りし，評価するか等の取り決めおよびその方法である．これはソフトウェア開発過程において，フェーズ毎に何をインプットとして，何を成果物としてレビューするのかを企業間で行うことに他ならない．

企業間で交換するこれらの情報を格納するのが CALS におけるリポジトリの役割である．CALS におけるリポジトリに格納される対象は，以下の 2 種類に分けられる．

1) 仕様書，要求書等の交換対象となる情報がどこにあるかを示すカタログ情報．

これは，どの企業に情報が存在するかを示すロケーション情報と目的とする企業内でどこに格納されているかを示す管理情報からなる．

2） 仕様書，要求書そのもの．

CALS アーキテクチャ[13]においては，1）の情報は，決められたあるサイトに格納される．このサイトは，利用者にあらかじめ知らされる．2）に関しては，物理的分散を許していて，1）の情報を管理するサイトが自動的に探して，目的となる情報を利用者に転送する場合と，1）の情報を基に，利用者が交換対象となる情報を直接参照する場合を想定している．

ここで，やり取りされる単位は物理的ファイル単位である．論理的単位，たとえば要求書全体であるのか，各節毎なのかは企業間の契約時に取り決めることとしている．したがって，論理的スキーマ構造は，企業間の取り決めに従い任意としている．このように，やり取りする企業毎にスキーマを定義する必要がありデータの統一は取れていない．

このような CALS アーキテクチャにおける問題点は以下の通りである．

1） 現状では，契約企業間ごとにスキーマを定義する必要があり，契約時の取り決めが重要となる．この労力を節約するためには，CALS に特化したある標準的なスキーマを規定する必要がある．必然的に管理する対象もある程度規定される．

2） 一つの発注に対して複数の企業が関わる場合，発注企業は，契約企業のリポジトリを条件等を指定して横断的に検索する必要も出てくる．このためには，ある程度スキーマを標準化して，情報を共有できる仕組みを作成しておくことが必要である．

3） 現状の CALS では文書の版管理を行なうかどうかは契約に任されている．したがって版管理を行なう場合，独自の仕組みを作成する必要がある．

このような問題点を解決する基盤技術としてのリポジトリ環境が最近整ってきた．それを以下解説し，今後の CALS 展望について述べる．

3. OMG（Object Management Group）の動向および業界の動向

本章では，リポジトリをめぐる標準化動向，業界動向について述べる．

3.1 OMG の動向[7][10][14][15][16]

OMG Meta Object Facility[12]は，リポジトリ・スキーマを定義する仕組みを規定した規格である．米国ユニシス社のオブジェクト指向リポジトリの UREP で提供されるリポジトリ・オブジェクト・モデルがそのベースとなっており，1997 年 11 月に標準規格として採用された．規格内容は，UREP の現在の実装を拡張したものとなっている．UREP は 1998 年中に規格に準拠する予定である．

ここで定義されたモデルは，5.2 節で述べるようにあらゆる分野のモデル定義に影響を及ぼす．OMG の他の活動，UML（Unified Modeling Language）定義および Business Object 定義においても参照され，利用されている．CORBA 準拠の様々な製品がこのモデルを何らかの形で取り込む必要があり，UREP は最初に実装する製品となる予定である．

3.2 業 界 動 向[11][12][20]

米国マイクロソフト社は，Visual Basic Enterprise バージョン 5.0 にバンドルして Microsoft リポジトリ 1.0 をリリースした．このリポジトリは，独立したリポジトリ

製品というよりは，Visual Basic のプロジェクト管理を第一義の目的として開発された．従って，Visual Basic で作成したプログラム情報は自動的に格納される．

Microsoft リポジトリの特徴は以下の通りである．

1) 格納された情報は OLE インタフェースを通してアクセスできる．
2) 情報モデルを提供して，任意のツールを統合できるよう設計されている．
3) データベースエンジンとしては，ODBC（Open Database Connectivity）インタフェースを備えている任意のデータベースをサポートする．従ってユーザのデータベースの選択の幅は広い．一方，データベースエンジンとして関係データベースを仮定しているため，複雑な関連および継承関係の管理には人工的な外部キーを使用しており，効率上問題が発生する可能性がある．

基本的には Microsoft リポジトリは，マイクトソフト社固有の技術（OLE，DCOM, ODBC 等）をベースにしており，オープン性には欠けるが，多くのベンダーが Microsoft リポジトリとのツール統合を表明しており今後の動向が注目される．

その他の製品として，PCTE に完全に準拠している点で，仏国ブル社の TranStar も注目される．

このように，標準化活動および業界における取り組みの両面で，リポジトリ関連の活動が活発化しており，CALS において広く認知され，かつ利用できる環境が整ってきている．

4. リポジトリ機能概要

リポジトリとして必要な機能は，様々な場でこれまで検討されてきた．ISO IRDS[8][9]，ANSI IRDS[4][5]，ECMA PCTE[2][3][6]，OMG 等標準化作業だけを見てもかなりの量になる．ただ残念なことに，これらの活動が広く認知されず，規格に準拠したプロダクトも少なく実際に使用されていないのが実状である．

これらの規格で検討・提案された機能の中から重要と思われるものを表 2 に示す．表内で使用するオブジェクトは，オブジェクト指向でいうオブジェクトではなく，管理対象となるリポジトリのデータという意味である．

5. UREP の機能および実装[17][18]

UREP は，どのような分野でも利用できるような汎用のリポジトリを目指した製品である．情報の拡張性および再利用性を考慮して，データ定義，API 等にオブジェクト指向技術を全面的に取り入れている．

リポジトリ機能は前章に述べたように，データベース管理システム（DBMS）の多くの機能をサポートし，DBMS の一つのアプリケーションとして実装すれば効果的である．この利点を生かして，UREP は現在 Versant オブジェクト・データベースのアプリケーションとして実装されている．API としては，C＋＋，C，OLE を提供している．

UREP の提供するリポジトリとしての機能は，情報モデルとリポジトリ・サービスに大別される．情報モデルは，OMG Meta Object Facility で定義されたメタメタモデルに基いて定義されている．リポジトリ・サービスは，ECMA PCTE で定義され

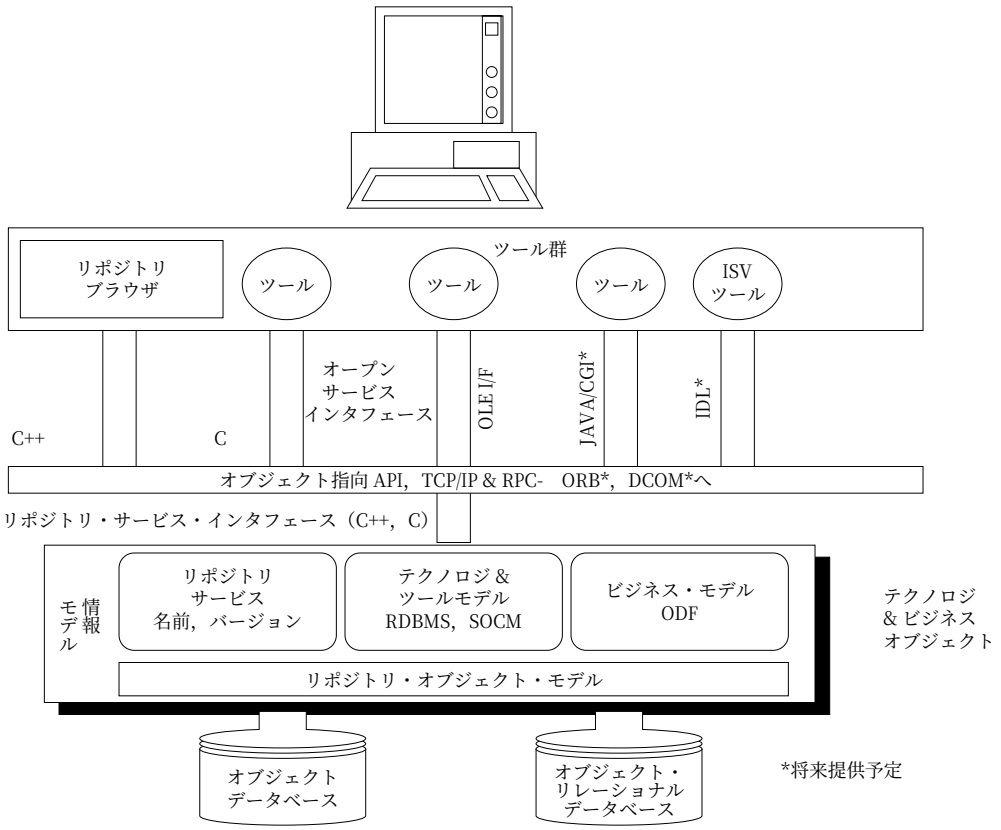
表 2　リポジトリの機能

リポジトリ機能	説　　明
スキーマ管理に関する機能	
スキーマ定義・管理	データ構造(スキーマ)を定義・管理する機能．スキーマ進化機能（スキーマを変更するとインスタンスも自動的に移行する機能）も含む．
メタデータ管理	標準的なモデルを提供して，リポジトリを利用するツール間でデータ交換を行なう機能．
参照整合性管理	オブジェクト間の制約・整合性を管理する機能．とくに，リポジトリではオブジェクトが複雑な関連を持つことが多く，システムにその制約・整合性を定義して，システムが管理する機能が必要となる．
パーティショニング	論理的なオブジェクトの集まりを管理する機能．DBMS のスキーマという概念に近い．
サブセッティング	オブジェクト群のサブセットを定義して，利用者にサブセットごとに権限をあたえる機能．CODASYL データベースのサブスキーマ，関係データベースのビューに類似している．
リポジトリ・サービスに関する機能	
アクセス・コントロール	モデルの定義・変更・削除および各オブジェクトに対するアクセス権限を管理，制御する機能．アクセス権限を管理するためのユーザ管理機能も含む．
問い合わせ機能	リポジトリ内のオブジェクトに対して問い合わせを行い，条件に合致するオブジェクトを検索する機能．
リモート・アクセス	ネットワークを利用して，透過的にリポジトリデータをアクセスする機能．
同時実行制御	ロック機構により，多数のユーザがコンカレントにアクセスしてもデータに不整合が生じないように管理する機能．データベース管理システム（DBMS）はこの機能をサポートしている．
構成管理	オブジェクトの集まりをひとつのオブジェクトと見なして管理する機能．
データ永続性	リポジトリに格納されたデータは,時間を超えて検索できる機能．DBMS の基本機能である．
トランザクション管理	リポジトリに対するアクセスの原子性を保証する機能．リポジトリに対する更新は，all or nothing であることを保証する．
名前サービス	オブジェクトに対して，利用者が認識できる文字列である名前を付与する機能．さらに，名前に関する制約（例えば，名前の長さなど）を定義・管理する機能．これにより，利用者は名前によりオブジェクトを検索できる．
バージョン管理	オブジェクトまたはオブジェクト群の異なるバージョンを作成・管理する機能．ソフトウェアの開発過程においては，しばしば変更を伴うため，バージョン管理を行って，ある時点の情報に戻る必要性もあり利用されている．
オブジェクト複写	オブジェクトまたはオブジェクト群の複製を作成する機能．これにより，利用者が不必要にデータを入力し直す必要がなくなる．
影響分析	オブジェクトの格納・変更・削除に伴って影響を及ぼすオブジェクト群をレポートする機能．プログラム管理等で利用される．
イベント通知	リポジトリ・アクセスに当たって，発生したイベントを利用者に通知する機能．

たサービスに基いている．

5.1　アーキテクチャ

UREP がサポートする機能は，ECMA で定義されたトースタモデル[3][6]のうちオブジェクト管理サービスおよびツールとオブジェクト管理サービスを繋ぐコミュニケーション・サービスの一部である．UREP のアーキテクチャを図 1 に示す．



ODF　: Organization Design Facility
SOCM : Software One Core Model
ORB　: Object Request Broker
DCOM : Distributed Component Object Model

図 1　UREP のアーキテクチャ

UREP は図 1 が示すように，DBMS の一つのアプリケーションとして実装されている．現在のバージョンでは，Versant オブジェクト・データベースを利用している．将来的には，ニーズに応じて，他の DBMS に対応することも可能である．

DBMS 上に，オブジェクト指向技術に基づいたリポジトリ・サービス・モデル（Repository Service Model：以下 RSM）を定義して，リポジトリ・サービスをメソッドとして提供している．ツールとリポジトリ間のコミュニケーションには RPC を使用しており，ツールからは透過的なアクセスを可能にしている．各ツールは，C＋＋，C または OLE インタフェースを通してリポジトリデータにアクセスできる．

5.2 情報モデル

リポジトリに格納される情報はモデルとしてその枠組みが定義される．モデルは，データベースにおけるスキーマに対応する．UREP では，ベースとなる情報モデルとして，RSM を提供している．

一般にモデルには，いくつかの階層があり，アプリケーション層も含めると図 2 のようになる．

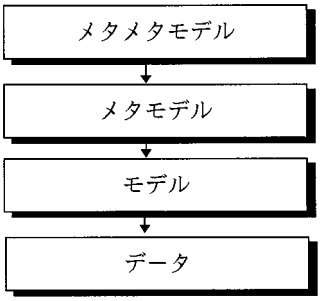


図 2　モデル階層

モデル階層において，上位の階層がその下のデータ定義情報を管理する．

メタメタモデル層は，このモデル階層の最上位に位置し，固定されたデータ構造を持つ．この階層は UREP では，RSM 内の UrepMeta という prefix のついたクラス群で構成され，リポジトリの構成要素の基盤を提供する．このモデルは，リポジトリ・オブジェクト・モデルとも呼ばれる．

メタモデル層には，CASE ツール等で扱う情報を管理するための定義情報が格納される．UREP では，リポジトリサービスを支える RSM のクラス群がこの層に属する．また，UREP の利用者は RSM のいずれかのクラスを継承する形で，独自のモデルを定義できるが，そのモデルもモデル階層の視点からみるとメタモデル層に属する．例えば，UML で扱う要素を管理するためのデータ構造は，この層に格納される．

モデル層には，CASE ツールにおけるユーザ情報等が格納される．例えば，UML を用いて定義したユーザのクラス情報や関係データベース設計ツールで生成されたユーザのデータベーススキーマ情報が格納される．UREP では，ほとんどのオブジェクト・インスタンスがこの層に属する．

最下位のデータ層は，ユーザが直接利用するデータが格納される．つまり，銀行の顧客情報や製品の部品情報が格納される．データ層は UREP の管理対象外である．

UREP を利用するアプリケーションは，上にも述べたように RSM に定義されたいずれかのクラスを継承したクラスを定義して，モデルを作成する．RSM のクラスを継承することにより，UREP が自動的に管理・維持する属性（例えば，オブジェクトの作成者や更新日時など）を利用したり，メソッドとして実装されているバージョン管理のための機能を利用することができる．

また，ツールでよく利用される，ファイル管理のための UrepFile クラス，ユーザを管理する UrepUser クラスや UrepUserGroup などを定義しており，ユーザが定義

する負荷を軽減している．例えば，リポジトリの使用者として管理者と従業員を管理対象とする場合，Manager クラスと Employee クラスを UrepUser のサブクラスとして定義することにより，リポジトリ利用者として必要な情報が管理される．

このように，RSM は，リポジトリ・サービスを提供するための情報管理という側

表 3　UREP がサポートする機能

リポジトリ機能	UREP における実装
スキーマ管理に関する機能	
スキーマ定義・管理	メタデータ・サービスとして実装. メタメタモデル（リポジトリ・オブジェクト・モデル）のインスタンスを操作することにより，モデルの追加・変更・削除ができる．
メタデータ管理	UREP　Exchange と呼ばれる別パッケージを通して CASE ツール間のデータ交換が可能．
参照整合性管理	メタデータ・サービスの一部として実装．モデル定義の際，様々な制約を定義できる．
パーティショニング	利用者の論理的な単位をモデルとして定義できる．これによりアクセス制御等が可能となる．
サブセッティング	UREP でのサポートは未定．
リポジトリ・サービスに関する機能	
アクセス・コントロール	ユーザ/セッションサービスとして実装．ユーザのカテゴリとしては，リポジトリ管理者，モデル所有者，ユーザの選択が可能である．オブジェクト単位のアクセス制御は未サポート．
問い合わせ機能	条件式によるオブジェクトの問い合わせ機能は提供していない．
リモート・アクセス	インタオペラビリティ・サービスとして実装．RPC を通して，アプリケーションは透過的にサーバ上にあるリポジトリ・データをアクセスすることができる．
同時実行制御	DBMS の機能を利用して実装．
構成管理	複合オブジェクトサービスとして実装．
データ永続性	永続サービスとして実装．RSM 内の UrepPersistentObject クラスを継承するクラスのオブジェクトはすべてデータベースに格納され，永続性が保証される．
トランザクション管理	トランザクション・サービスとして実装．バージョン管理の単位となるロングトランザクション，データ更新およびロックの単位となるショートトランザクション，サブタスクを制御する入れ子トランザクションをサポートしている．
名前サービス	名前サービスとして実装．利用者は名前付け規則・制約が定義された名前空間を指定して，オブジェクトに対する名前付けおよびオブジェクトの検索ができる．
バージョン管理	バージョン・サービスとして実装．オブジェクトのバージョンおよびバリアントを作成・管理することができる．
オブジェクト複写	UrepPersistentObject クラスのメソッドとして実装．
影響分析	次期バージョンで提供予定．
イベント通知	診断サービスとして実装．リポジトリアクセスにおけるイベントは，通知スタックを通して検知できる．

面と利用者が再利用できるように共通の枠組みを提供する側面を持つ．

5.3 リポジトリ・サービス

4 章で示したリポジトリ機能に沿って，現段階で UREP が実装している機能を検証したものが表 3 である．

影の部分が UREP に実装されていない機能である．UREP はほとんどの機能を実装しておりリポジトリ製品としての完成度は高い．この中で，UREP の特長となっているバージョン・サービスおよび複合オブジェクト・サービスについて詳しく説明する．

5.3.1 バージョン・サービス

UREP で管理されるオブジェクトのうち，UrepVersionedObject クラスを継承するクラスのオブジェクトはバージョン付け可能である．オブジェクトのバージョンは，バージョン番号とバリアント名によって管理される．バージョン番号は，オブジェクトの変更履歴を管理するために使用される．一方，バリアント名は並行開発のために，あるオブジェクトのコピーを作成して開発を進める場合，並行して進化するラインを管理するために使用される．

また，一度分岐したラインをマージする機能や途中のバージョンを削除して，“ バージョン・ツリー ” を整理する機能も提供されている．

バージョンサービスでは，作成可能なバージョンおよびバリアントの最大幅を設定することができる．これにより，不要なバージョン作成を抑制することができる．

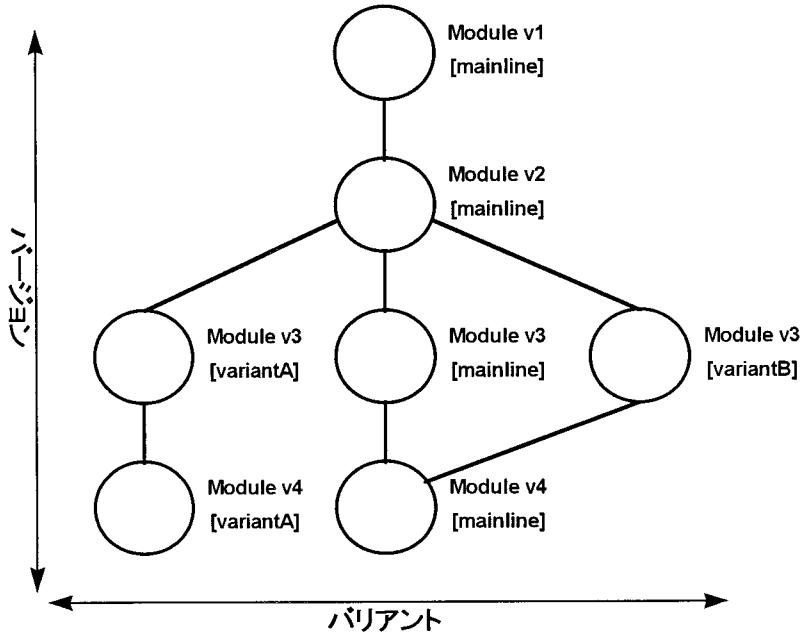


図 3 バージョン管理

図 3 において，Module というオブジェクトを作成するとまずバージョン 1（v 1）が mainline 上に作成される．バージョンを上げる操作を行なうと，mainline 上のバ

ージョンが，v 2，v 3…と進む．一方，バリアント名を指定して（ここでは，A，B 等を指定している）新しいバージョンを作成すると，mainline から分岐したルート上に新たなオブジェクトが作成される．

5.3.2 複合オブジェクト・サービス

複合オブジェクトは，オブジェクトの集まりを一つのバージョン付け可能なオブジェクトとして扱う機能である．このとき，個々のオブジェクトをコンポーネントと呼ぶ．複合オブジェクトは，他の複合オブジェクトをコンポーネントとして含むことができ，複合オブジェクトの階層を形づくることができる．複合オブジェクトの階層は，トップ・オブジェクトを指定することにより識別される．

具体例で見てみよう．図 4 において，Customer ファイルのバージョン 1 は，Address レコードのバージョン 1 をコンポーネントとして含む．同様に，Address レコードは，Name フィールド，City フィールド，ZipCode フィールドのバージョン 1 をコンポーネントとして含む．ここで，Customer ファイルをトップオブジェクトと指定するとこれらすべてのオブジェクト群が，複合オブジェクトとして認識される．したがって，ZipCode フィールドを変更してバージョン 2 が作成されると，それに伴って Address レコードも変更が自動的に認識されバージョン 2 が作成される．この Address レコードは，バージョン 1 の Name，City とバージョン 2 の ZipCode から構成される．Customer ファイルに関しても同様な変更が行われる．新しいバージョンの作成は，トップオブジェクトに辿り着くと終了する．これがトップ・オブジェクトが複合オブジェクトを認識するという意味である．

このように，複合オブジェクト・サービスは，複雑なオブジェクトのバージョン管理を容易にする働きを持つ．

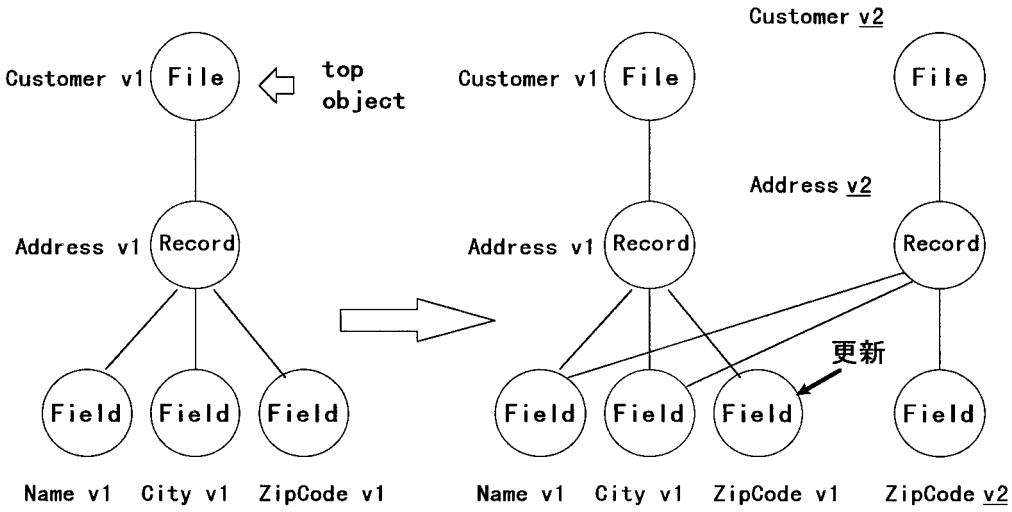


図 4　複合オブジェクトのバージョン管理

6. UREP の適用事例

UREP のアプリケーションとしては，一般的に CASE ツールが多いが，ここでは

英国 Adaptive Solution for Business 社が開発した Adaptive フレームワークと呼ばれる，ビジネス情報を管理するパッケージを紹介する[1][23]．

Adaptive フレームワークは，組織内および組織間に亘るビジネスプロセス，人，システム，オペレーション間の複雑な関連を表現し，意思決定に役立てるためのフレームワークを提供する．これにより，組織編成や変更およびビジネスプロセスを管理し，組織やビジネスプロセスの変更に伴う影響を分析することができる．また，組織内で進めているプロジェクトを管理して組織としての目的達成を支援する．

Adaptive フレームワークは，ビジネス情報モデルにより，組織が管理を必要とする領域を定義している．このモデルは，組織の要求に応じてカスタマイズ可能である．

プロダクトのアーキテクチャ的には，全面的にオブジェクト指向技術を採用している．

この中で，UREP はオブジェクトを格納するリポジトリとして利用され，動的に変更される，複雑な関連を持つ情報を管理する．また，変更履歴はバージョン機能を用いて管理される．このプロダクトの中でオブジェクト管理に必要とされる要件は，

1） C＋＋によりアクセスできること
2） オブジェクトを属性で識別して操作できること
3） バイナリ・ラージ・オブジェクトを格納・管理できること
4） インスタンスを操作するのと同じ API でメタモデルを定義・変更できること
5） オブジェクトのバージョンを管理できること
6） トランザクション管理の下にオブジェクトを操作できること

である．

UREP はこれらの要求をすべて満たし，Adaptive Solution の中核ソフトウェアとして稼働している．

7. UREP と CALS

2 章で CALS の問題としてあげたもののうち，標準的なスキーマ定義をするためのベースとして，UREP では RSM を提供している．また，文書の版管理も UREP のバージョン管理機能により容易に実現可能である．さらに，データ定義の変更も，メタデータ・サービスにより，API を通して可能であり，拡張性に優れている．このように，CALS における問題点を解決するための環境は整っている．

今後 CALS における標準スキーマを定義することが最も重要なことである．その際，企業間のデータ交換の粒度，つまりファイル単位で行なうのか節単位で行なうのかの違いを容易に管理できるように工夫する等，現状の問題点を細かく分析した上で，スキーマを定義する必要があり，大きな課題である．

8. 今後の展望

CALS の要求を満たすリポジトリ機能は，標準化活動および実装の両面で実用化段階に入りつつある．これを踏まえて，CALS アーキテクチャを拡張し標準化されることが望まれる．アーキテクチャの標準化・流布により，CALS が広く利用され，リポジトリの重要性が認識されることを望みたい．

参考文献
［1］ Adaptive“ Eugene - A Visual Object Management Framework ”, UREP Evaluation Kit.
［2］ 鯵坂恒夫, 沢田篤史, 満田成紀 “ Emeraude PCTE ” コンピュータソフトウェア, Vol. 10 No. 2, 1993, pp. 65〜77.
［3］ 鯵坂恒夫 “ 開放型 CASE プラットフォーム ” コンピュータソフトウェア, Vol. 10 No. 2, 1993, pp. 4〜12.
［4］ ANSI “ Information Resource Dictionary System（IRDS）Reference Model-X 3 H 4.1 ” May 10, 1991.
［5］ ANSI “ Information Resource Dictionary System Requirement Specification ” August 25, 1992
［6］ ECMA（European Computer Manufacturers Association）“ Reference Model for Frameworks of Software Engineering Environments, Technical Report ECMA TR/55 2 nd Edition ” NIST Special Publication December 1991.
［7］ IBM“ IBM response to OMG RFI#3 ”OMG document tc/95-11-09, 1995.
［8］ ISO/IEC 10027 “ Information Resource Dictionary System（IRDS）Framework ” 1990.
［9］ ISO/IEC 10728 “ Information Resource Dictionary System（IRDS）Service Interface ”1992.
［10］ OMG “ Meta Object Facility（MOF）Specification ” OMG document ad/97-08-14, September 1, 1997.
［11］ “ Patrica Seybold Group’s SnapShots ” August 1995.
［12］ Paul Harmon“ OO Repositories ”Object Oriented Strategies, January 1998.
［13］ 日本規格協会 “ ソフトウェア CALS 標準化調査研究委員会報告書 ” 1996.
［14］ Rational, Microsoft, Oracle, Unisys et al.“ Proposal to the OMG’s Analysis and Design Task Force - Part I UML Definition version 1.1 ”September 1, 1997.
［15］ Texas Instruments“ Texas Instruments response to OMG RFI#3 ”OMG document tc/95-11-06, 1995.
［16］ Unisys“ Unisys response to OMG RFI#3 ”OMG document tc/95-11-05, 1995.
［17］ Unisys “ Universal Repository Capabilities Overview for ISVs – Embedding the Unisys Object Repository in Resale Commercial Solutions ”1996.
［18］ Unisys“ Universal Repository Technical Overview ”1996.
［19］ Adaptive Web : http : //www.adaptive-solutions.com.
［20］ Microsoft Web : http : //www.microsoft.com/repository.

執筆者紹介　行　成　　敦（Atsushi Yukinari）
1961 年生．1986 年東京大学大学院理学系研究科修士課程（数学）修了．同年日本ユニシス(株)に入社．シリーズ 2200 DBMS の開発・保守に従事. 1995 年より UREP 担当．プラネットフォームビジネス部データベースソフトウェア室所属．ソフトウェア CALS WG 6（基盤アーキテクチャ）委員.