

## 3Z-3 大規模開発支援向け OMG™ MOF™ 準拠リポジトリの試作と評価\*

千葉俊哉<sup>†</sup> 吉野松樹<sup>††</sup> 藤井啓詞<sup>††</sup> 黒井充<sup>††</sup> 荒井仁<sup>††</sup>

(<sup>†</sup>:(株)日立製作所システム開発研究所, <sup>††</sup>:(株)日立製作所ソフトウェア事業部)

### 1. はじめに

ソフトウェアの設計情報を細粒度のオブジェクト群として版管理することのできる、ソフトウェア開発支援向けリポジトリを開発している。本稿では特に、大規模開発向け機能を実現するにあたっての技術的問題に対する解決案と、試作による本解決案の評価結果の概要をまとめる。

### 2. 大規模開発支援向けリポジトリの要件

まず前提として、大規模開発支援向けリポジトリの要件を明確にするため、我々が試作に先立って行った要件ヒアリングの内容をまとめる。ヒアリングは下記3製品の開発チームとの、6ヶ月前後のディスカッションの形でいった。

- (1) UML™設計ツール:大規模オブジェクト指向開発に対応するために、複数開発者による共同作業での UML オブジェクトモデル設計を狙いとする。
- (2) ワークフロー作成ツール:ワークフローを構成する部品を蓄積しておき、これらを再利用しながら組み合わせて定義を行う。
- (3) データウェアハウス設計ツール:既存データベースから吸い上げるなどした物理テーブル設計情報から必要なテーブルカラムを拾い出し、これを新規に作成する論理テーブルにマッピングする。また、後の保守/運用において、物理テーブルと論理テーブルとの間の依存関係をテーブル単位で確認しながら運用する。

ここではヒアリング結果の詳細は割愛するが、得られた要件の中で重要なものをまとめると、次の4点に集約される。

- (A) インターフェースの規格化:リポジトリへの設計情報の出し入れのインターフェース仕様が明確に規格化されていること。
- (B) 下位互換性を保った拡張性:ツールのエンハンスに伴い、リポジトリに格納する設計情報のスキーマを、下位互換性を保持しつつ拡張できること。
- (C) 並行作業と相互反映:大規模な設計情報を複数作業で共同作成・改訂するために、一つの情報要素を複数作業で共有しながら作業したり、全体を細かく分けて分担したりできること。
- (D) 細粒度の変更波及分析:仕様書単位ではなく仕様書の部分単位で変更した場合の影響波及を検知できること。

大規模開発支援への実適用に耐えるリポジトリとするためには、以上の要件を、現実的なレスポンス性能が得られる方

式で実現しなければならない。

### 3. 方式概要

2章にまとめた各要件を満たすためにとった方式の概要を以下にまとめる。

#### 3.1. Meta Object Facility (MOF) への準拠

要件(A)を満たし、かつ将来の異ベンダツール間データ交換の可能性を視野に入れるため、国際標準化団体Object Management Group (OMG)のMOF規格[1]に準拠する方針をとることとした。MOFは、メタデータ管理向け汎用リポジトリのインターフェース規格である。1997/11にバージョン1.1が採択済、現在の最新改定版は1.3である。

MOFは特定用途にとどまらない、OMGの共通リポジトリ基盤としての規格化を狙いとしている。このため、格納するメタ情報の種類を特定のものに限定したリポジトリインターフェースを規格化するのではなく、「リポジトリで管理するメタ情報のスキーマ」から「リポジトリインターフェース(実体はIDL)」へのマッピングを規格化している。また、OMGでは上記MOFに基づくメタデータのスキーマからXMLへのマッピング規格をXMI™[2]として規格化している。UML[3]、CWM™[4]など、OMGのメタデータ規格は、MOF規格に基づいて規格化され、その交換形式はXMIに基づく方向に向かっている。

我々の試作ではMOF準拠のスキーマ定義を可能とし、インターフェースもMOFに合わせる方針を採ることとした。版管理機能など、MOF規格で規定されていない機能のインターフェースは独自仕様とした。

#### 3.2. スキーマの継承と付加情報による2段階の拡張性

MOF規格ではスキーマの定義にオブジェクト指向の「継承」を利用できる。この「継承」を用いることで既存スキーマとの互換性を維持しつつ、スキーマを拡張することができる。また、例えばダイアグラムの座標、サイズ、フォントの情報など、あえてリポジトリのスキーマとしてメンテナンスする必然性のない情報を格納できるように、リポジトリの各オブジェクトにツール独自の付加情報を格納できる仕組みを持たせた。以上「スキーマ継承」と「付加情報」の2段階の拡張性を与えることで、要件(B)に対処した。

#### 3.3. オブジェクトグループの版管理

一枚の仕様書を、ファイルなどの一枚岩のデータで管理すると、要件(C)、(D)を満たすことができない。要件(C)を満たすためには共有や分担の単位に、また要件(D)を満たすためには変更波及分析の単位に、それぞれ細分化した、細粒度のオブジェクトで管理することが必須となる。

また、要件(C)を満たすためには、出し入れ/変更される

\*Prototyping and Evaluation of OMG™ MOF™ Compliant Repository for System Developers

<sup>†</sup>Toshiya Chiba      Systems Development Lab., Hitachi, Ltd.

OMG marks and logos are trademarks or registered trademarks, service marks and/or certification marks of Object Management Group, Inc. registered in the United States.

オブジェクト単位の排他管理が可能で、かつ仕様書ごとに版管理されているように扱える機能が必要である。その素朴な実現方式としては、まずリポジトリ内のオブジェクト一つ一つを版管理し、さらに仕様書単位などにオブジェクトをまとめて構成管理をする方法が考えられる。しかし、この方式ではオーバーヘッドが非常に大きくなるという問題がある。本方式に基づく性能単価測定とそれに基づく試算を行ったところ、例えば100個のRDBテーブル設計から成るRDB設計仕様書をリポジトリから取り出す操作に1298.2秒(約21分)かかるという結果を得た(表 2)。これでは、中～大規模開発支援で実用的なレスポンスを得る事は難しい。

ところが、ここで再び要件(C), (D)を実適用の場面に照らし合わせて見直すと、細粒度の排他、変更相互反映、変更波及分析はいずれも最新の版のみに必要な機能であることがわかる。過去の履歴情報は、過去の時点の情報である以上、それらが後から変更されることはないためである。このことに着目し、改善を行った後の方式概要を図 1に示す。

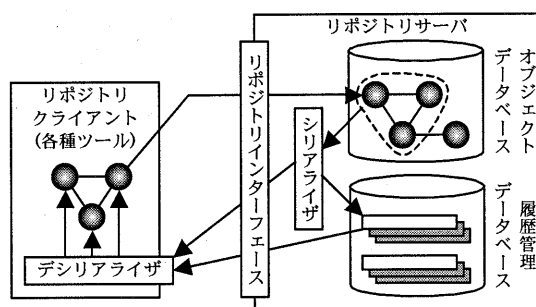


図 1: 方式概要図

リポジトリには最新版のオブジェクトを管理するオブジェクトデータベースと、グループ単位のオブジェクト群履歴を保存する履歴管理データベースの、2系統のデータベースを持たせる。リポジトリに対する操作は表 1の5種類とする。

表 1: リポジトリの操作一覧

|         |                                                                                                                     |
|---------|---------------------------------------------------------------------------------------------------------------------|
| 新規登録    | ツールで新規に作成した設計情報をリポジトリに登録する操作。オブジェクトデータベースに対応するオブジェクトを追加する。                                                          |
| チェックアウト | リポジトリ内の設計情報を変更する目的で取り出す操作。グループに属するオブジェクトに排他をかけて取得する。                                                                |
| チェックイン  | チェックアウトしたオブジェクトへの変更をリポジトリに反映し、変更作業を終了する操作。グループに属するオブジェクト群をオブジェクトデータベースから履歴管理データベースに保存した上で、オブジェクトデータベースを更新し、排他を終了する。 |
| 最新版参照   | オブジェクトデータベースから、グループに属するオブジェクト群を取得する。                                                                                |
| 旧版参照    | 履歴管理データベースに保存されているオブジェクト群の情報を取得する。                                                                                  |

以上の構成によりオブジェクトデータベースには常に最新版のオブジェクトが格納され、履歴管理データベースには変更履歴がオブジェクトのグループ単位で保存される。

#### 4. 試作による方式評価

上に述べた方式に基づくプロトタイプを開発し、方式の妥当性評価を行った。評価のポイントは、要件の満足度、レスポンス性能、ディスク消費容量、の3点である。以下、評価の

結果をまとめる。

- (1) 要件の満足度: 各ツールの機能仕様書・検討メモに基づくユースケースの机上評価と、ヒアリングによる確認により、要件を満足できる見込みであることを確認した。
- (2) レスポンス性能: 100 個のテーブル設計情報から成る RDB 設計情報に対する各種操作を行った場合のレスポンス特性(処理の終了までにかかる時間)を表 2 にまとめる。「新規登録」「チェックイン」は本方式による場合も処理時間がかかるものの、いずれも更新系の操作であるため、キューによる遅延処理を導入することで、たとえ大規模な仕様情報のケースでも十分実用範囲内で必要機能が実現可能といえる。
- (3) ディスク消費容量: 上記同様の 100 個のテーブル設計情報から成る RDB 設計情報を保存すると 1 履歴あたり約 600KB のファイル容量を消費した。これは十分に現実範囲内の値といえる。変更差分のみを保存することで、ディスク消費容量はさらに削減可能であろう。

表 2: レスポンス性能一覧

| 操作内容    | 改善前*        | 改善後        |
|---------|-------------|------------|
| 新規登録    | 2469.0 Sec. | 190.0 Sec. |
| チェックアウト | 1298.3 Sec. | 9.2 Sec.   |
| チェックイン  | 680.9 Sec.  | 69.0 Sec.  |
| 最新版参照   | 120.3 Sec.  | 6.5 Sec.   |

\*改善前: オブジェクト毎に版管理し、グループ毎に構成管理した場合

#### 5. まとめ

大規模開発支援向けリポジトリに求められる要件を実製品の開発部隊にヒアリングし、その結果に基づいて要件の満足と性能を両立する方式を設計した。試作による方式評価を通し、本方式が実際に要件と性能を両立できる見込みであることを確認した。細粒度リポジトリの試み自体は既にいくつか行われているが[5][6]、本稿で述べた方式では大規模共同設計向けに細粒度オブジェクトをグループ化した版管理方式を検討し、その実現性を実証済みである点が、従来にない特長である。

#### 参考文献

- [1] Meta Object Facility (MOF) Specification Version 1.3, OMG Document formal/00-04-03, 2000
- [2] XML Metadata Interchange (XMI) Specification Version 1.1, OMG Document formal/2000-11-02, 2000
- [3] Unified Modeling Language Specification Version 1.3, OMG Document formal/2000-03-01, 2000
- [4] Common Warehouse Metamodel (CWM) Specification, OMG Document ad/2000-01-01, 2000
- [5] 福安直樹, 山本晋一郎, 阿草清滋: 細粒度リポジトリに基づいた CASE プラットフォーム Sapid, 情報処理学会論文誌, Vol.39, No.6, 1998
- [6] 近藤竜生, 大橋恭子, 中山裕子, 吉田裕之: 業務仕様の共同作成を支援するリポジトリ, オブジェクト指向'95 シンポジウム論文集, pp.317-324, 1995