ソフトウェア工学 115－2
（1997. 7. 31）

メタ階層アーキテクチャによるモデリングとリポジトリシステム

小飼 敬　　中野 喜之　　上田 賀一

茨城大学 工学部 情報工学科
〒316 茨城県日立市中成沢町 4-12-1

本報告では，ソフトウェアモデルにおけるメタ階層を定義し，これを基盤としたモデリング環境と，この環境を対象としたリポジトリシステムを構築する．メタ階層でモデルの役割を各レベルに分割することで、広範囲なソフトウェアドメインのモデリングを支援できる。これによって、ラピッドプロトタイピングとエンドユーザコンピューティングの環境を実現し、ソフトウェア開発における開発者と要求者間の思考の隔たりを埋める。リポジトリシステムは、成果物をドキュメント化することで保守や再利用時に把握しやすくし、開発プロセスを支援することで高品質な成果物の作成を促進する。更にモデリング環境を含むリポジトリシステムは，分散した環境においても相互に通信できる機能を持つため、ソフトウェアリポジトリを中心とした分散開発を可能とし、近年の開発環境の形態をサポートする。

Modeling Environment based on Meta Hierarchical Architecture and Software Repository System for its Environment

Kei Kogai　　Yoshiyuki Nakano　　Yoshikazu Ueda

Ibaraki University
4-12-1 Naka-Narusawa, Hitachi, Ibaraki, 316 Japan

In this paper, we presents a meta hierarchy, a modeling environment based on this hierarchy, and a repository system for this environment. In meta hierarchy, separating a role of models into three levels, it is possible to support model development on various domain. Thus, a rapid prototyping and end-user computing environment is realized and a mutual understanding between developers and requirers is built up. The repository system supports a documentation of products for comprehensibility, a navigation on modeling process for good-quality, and a distributed development environment for the style of current software development.

1 はじめに

大規模開発において開発者と要求者の意志の疎通を図ることが重要である．これを行うには，開発者側がシステムのプロトタイプを素早く作成し，要求者からのフィードバックによって，システムを洗練していくラピッドプロトタイピングと，要求者が直接開発に参加していくエンドユーザコンピューティングの二つの方法がある．既存のCASE環境には，プロトタイピングを支援するものも存在するが，それらの殆どは，状態遷移図などのシステムのある側面のみのプロトタイピングを支援するように特化されている [4]．また，4GLなどエンドユーザコンピューティングを支援するための環境や言語も存在するが，それらは，特定のドメインに特化されているものが多く，実際に使用するには制約が大きい．

以前より我々はメタ階層に基づくモデル構築支援環境を開発してきた [1] が，そこではERモデルで表すことのできるモデルしか記述できないなどの不十分な点があった．そこで本報告では，ソフトウェアモデルのメタモデルを定義し直し，広範囲のソフトウェアドメインをカバーすることを可能とするため，メタモデルを記述するためのメタモデルを定義し，このメタ階層を基盤とし，柔軟なプロトタイピングとエンドユーザコンピューティングを可能とするモデリング環境を構築する．

一方，ソフトウェア開発において生成される成果物は，以降の開発フェーズや別のソフトウェア開発に利用するためにソフトウェアリポジトリによって永続性が保証されたまま管理される必要がある [6]．このことはモデリング環境においても同じことが言え，モデリング過程に対する成果物を管理する必要がある．そこで本報告では，上記のモデリング環境に適応させたリポジトリシステムについても考察する．リポジトリシステムは，システム利用者が成果物を把握しやすくするために成果物をドキュメント化して管理し，品質の良い成果物を作成するために開発プロセスを支援する．

本報告では，前半にメタ階層アーキテクチャとそれに基づくモデリング環境について述べ，後半にこの環境を対象としたリポジトリシステムについて述べる．

2 モデリング環境

モデリング環境では，モデルの生成や編集，修正を行うことができる．本環境で扱う基本的なプロダクトは，モデルである．本環境とそのプロダクトであるモデルは，オブジェクト指向モデル記述言語 Bramble[2] によって記述される．

2.1 モデル–メタモデル

本環境は，モデルの記述と視覚化を目的としている．モデルとは，現実世界のある側面を記述したものであるが，モデルには，その目的によってさまざまな種類が存在する．例えば，データフローモデル，状態遷移モデル，ERモデル，ペトリネットモデルなどがある．そして，一般的なモデルより，あるドメインに特化したモデルの方が，記述効率が高いと言える．本環境では，このようなさまざまなドメインのモデルを統合的に扱うことができるよう，モデルの“メタモデル”を次のように定義した．

メタモデルは，モデルを記述し，解釈する．

メタモデルは，モデルを記述するためのモデラを持つ．このモデラは，メタモデルの定義に基づき，モデルを視覚的にモデリングできる．

2.2 メタ階層アーキテクチャ

メタ階層アーキテクチャでは，メタモデル開発をより容易にするために，メタモデルを記述するためのメタモデルである“メタメタモデル”を定義し，モデルのメタ階層を，ベースレベル，メタレベル，メタメタレベルの3つの階層に分割した．これにより，この階層が，モデル–メタモデルという関係で一貫して記述されることになる．

このような分割により，モデル記述者は，適切なドメインのメタモデルを選択することによってドメイン固有の問題の記述に専念することができる．また，メタメタモデルを用意することで，ドメインエキスパートによるドメイン固有のメタモデル構築を助けることができる．

以下に，メタ階層アーキテクチャの各階層を説明する．

2.2.1 メタメタレベル

メタメタレベルには，4つのメタモデルMMM, FIELD, ENTITY, RERATIONSHIPが存在し，これらをまとめてメタメタモデルと呼ぶ．

MMMは，メタレベルのモデルを記述するメタモデルである．MMMは，FIELDによって記述される．FIELDによって記述されたメタレベルのモデルは，FIELD, ENTITY, RELATIONSHIPによって記述されるメタレベルのモデルを集約したモデルを記述できる．FIELDは，ENTITYによって記述される．ENTITYは，“プリミティブなモデルを記述するためのメタレベルのモデル”を記述するためのメタモデルである．ENTITYは，ENTITYによって記述される．RELATIONSHIPは，メタレベルのモデルにおけるモデル間の関係を記述するためのメタモデルである．RELATIONSHIPは，ENTITYによって記述される．

また，メタメタモデルのメタモデルは，メタメタモデルである．

メタメタモデルは，メタレベルのモデルを，FIELD, ENTITYによって記述されるメタモデルをノードで，RELATIONSHIPによって記述されるメタモデルをアークでER図のように視覚化する．また，これらのER図は，FIELDによって階層化することができる．このメタメタモデルによって記述されたメタメタモデルを図1に示す．

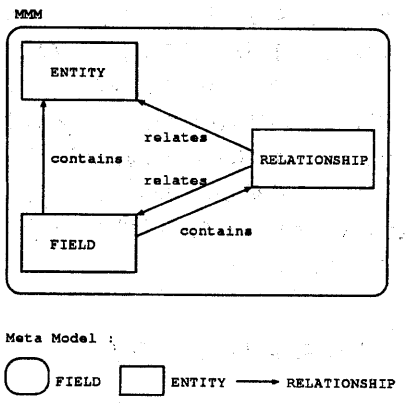

図 1: メタメタモデル

2.2.2 メタレベル

メタレベルのモデルは，モデルを記述する．また，メタレベルのモデルは，メタメタモデルによって記述される．

基本的に，メタレベルのモデルは，MMMのモデラによって記述され，ベースレベルモデルを記述するためのモデラを持つ．モデラは，ベースレベルのモデルのドメインに応じたモデリング方式を取ることができる．例えば，ERモデルなどはNA（ノード・アーク）モデラが最適であるし，オブジェクトモデルなどは継承や集約を表現できる独自のモデラを持つことができる．

2.2.3 ベースレベル

ベースレベルのモデルは，メタレベルのモデルによって記述される．

通常，要求者が求める成果物は，このレベルのモデルとなる．前述の通り，メタレベルのモデルはそのドメインに応じたモデラを持っており，また，ドメイン固有の情報を含んでいるので，ベースレベルのモデル作成者は，視覚的で，より直感的なモデル構築が可能となる．

2.3 モデルコンポーネント

メタレベルのモデルの開発効率を上げるために，メタメタモデルを定義したが，実際には，それだけでは容易にメタモデルを開発できるとは言えない．そこで，あらかじめ汎用的なメタモデルをコンポーネントとして作成しておき，これらを修正し組み合わせることによって，新たなメタモデルを作成できるようにした．同様に，ベースレベルにもコンポーネントを作成することができる．もちろん，新たに作成したモデルもコンポーネントとして再利用できる．現状では，メタレベルコンポーネントは表1のようなものがある．

これらは，全てメタメタレベルモデル“FIELD”によって記述されたモデルである．また，以前作成されたBrambleの視覚的開発環境であるVisualBramble[3]も，コンポーネント化することによって本環境に統合される予定である．

ここで，メタレベルコンポーネントの詳細な定義の例として，データフローモデルを取り上げる．デ

表 1: モデルコンポーネント

モデル名	説明
データフローモデル 状態遷移モデル	一般的な目的に使用されるメタモデル
スクリプトモデル	モデルの全体的な流れ（メッセージや制御）を制御するためのメタモデル
ウィジェットモデル	GUI の定義を行うためのメタモデル
ガイドモデル	モデリングのナビゲートを行うためのメタモデル

ータフローモデルのメタメタモデル（MMM）による図式表現を図 2 に示す．メタレベルモデル “データフローモデル” は，7 種類のコンポーネントから構成され，それぞれ，DataFlowModel は FIELD によって記述され，データを運ぶ DataFlow，データを格納する DataStore，データを変換する Process，データを生産/消費する Actor は ENTITY に，データの流れる方向を表す from，to は RELATIONSHIP によって記述される．

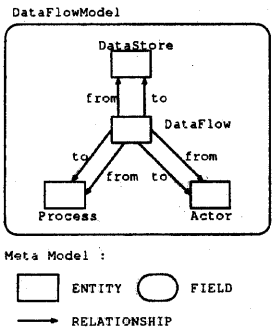

図 2: データフローモデル

2.4 モデル作成例

モデルの例として，メタレベルモデルのデータフローモデルとウィジェットモデル（図 3）を使用して，文章データベースを検索し，その結果を表示するモデルを作成する．このモデルでは，二つのベースレベルモデル，データフローモデルとウィジェットモデル（図 4）が作成される．

ウィジェットモデルでは，このモデルのインタフェースが定義される．具体的には，検索を行うための文字列を入力するテキストボックス，新規検索/絞り込み検索を選択するラジオボタン，検索開始ボタン，検索結果を表示するリストボックスで定義される．

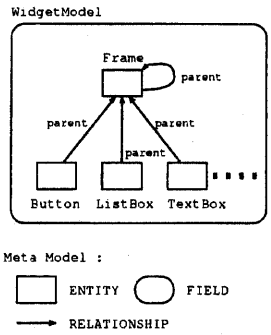

図 3: ウィジェットモデル

データフローモデルは，Actor によって記述されるものとして，入力文字列，検索オプション，検索開始フラグ，検索文字列，結果表示と，DataStore によって記述されるものとして，検索文字列，文書データベースと，Process によって記述される検索エンジンから構成され，それらの間が DataFlow によって記述される矢印によって結ばれる．

データフローモデルとウィジェットモデル中の，入力文字列とテキストボックス，検索オプションとラジオボタン，検索開始フラグと検索開始ボタン，結果表示とリストボックスはそれぞれ対応関係にある．例えば，ウィジェットモデルのテキストボックスに文字列が入力されれば，データフローモデルの入力文字列からその文字列が流れ，また，データフローモデルにおいて，検索エンジンから結果表示へ間作結果が流れれば，ウィジェットモデルのリストボックスにその検索結果が表示される．

このように，複数のモデルを作成し，それらを関連付けることによって全体的なモデルを作成することもできる．

3 リポジトリシステム

3.1 リポジトリシステムの目的

リポジトリシステムの主目的は，メタ階層に基づいたモデリング環境を対象とした成果物の管理

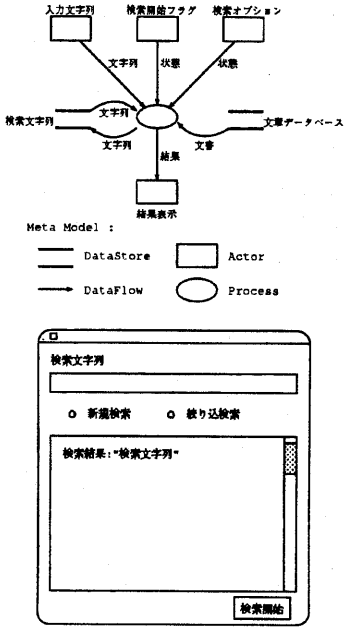

図 4: データフローモデルとウィジェットモデル

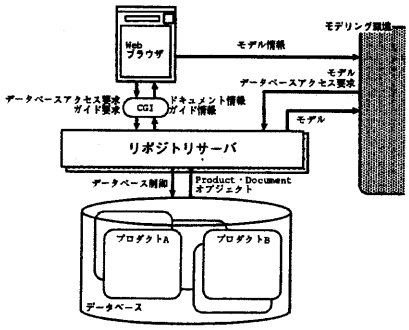

図 5: リポジトリシステム概観

である．従ってリポジトリシステムが扱う成果物は，このモデリング環境上で生成されたモデルとなる．モデリング環境利用者が生成・編集するのはメタレベルとベースレベルのモデルであるため，リポジトリシステムで扱うモデルはこの 2 つのレベルのモデルとなる．このようにリポジトリシステムで扱う成果物の形式は全てモデルで統一される．

これに加えてリポジトリシステムは，今日のソフトウェアリポジトリに必要とされるプロダクトの永続化とドキュメント化，開発プロセスの支援をサポートする．

3.2 リポジトリシステムの概観

リポジトリシステムの概観を図 5 に示す．

まずプロダクトを格納するためのデータベースがある．データベースに格納されたプロダクトは永続性が保証され，リポジトリサーバによって管理される．データベースにはオブジェクト指向データベースを用いる．本システムの中心となるリポジトリサーバは，システム利用者のリポジトリに対する要求に応えるサーバである．システム利用者は，Web ブラウザもしくはモデリング環境からリポジトリサーバに要求を送る．Web ブラウザは，システム利用者がモデリング環境を利用する際に，まず最初にモデリング環境を起動するのに必要となるプロダクトを取り出すための手段として利用する．システム利用者は Web ブラウザを使って，データベース内をブラウズすることができる．

3.3 リポジトリサーバ

リポジトリサーバが提供する機能は，大きく分けてプロダクト管理とプロセス管理の 2 つである．以下にリポジトリサーバの各機能について述べる．

3.3.1 プロダクト管理

リポジトリサーバはプロダクトをデータベースに格納する際に，ドキュメント情報を付け加えて格納する．データベース内のプロダクトに対するバージョン管理はモデル単位で行い，システム利用者がプロダクト格納の際に明示的に指定することでバージョンを付ける．ドキュメント情報が付加されて格納されるプロダクトは，図 6 に示す構造になる．各構成要素は以下のようになる．

Document オブジェクトはリポジトリシステムが付加するドキュメント情報を属性として持つ．Document オブジェクトはの次の属性を持つ．version はシステム利用者がプロダクトに明示的に付けるバージョン名，author はプロダクトを作成したユーザ名，date はプロダクトを作成した日付，comment はプロダクトに対する説明，model は実

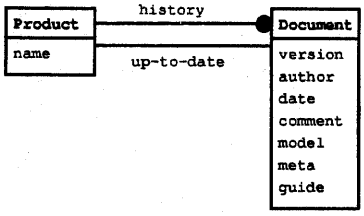

図 6: プロダクトのオブジェクト図

際に作成したモデルに対する識別子，meta は属性 model のメタの概念となるモデルに対する識別子，guide は属性 meta のモデルが持つモデラに対するガイドモデルの識別子である．属性 meta に対応するモデルがメタメタモデルになる場合は，これはモデリング環境で最初から提供されるモデルであるため，属性 meta は値を持たない．

Product オブジェクトは概念的にはシステム利用者がデータベースに対して扱える基本単位となる．Product オブジェクトの属性には name があり，これはシステム利用者がプロダクトに付けるユニークな名前である．

Product と Document は，history と up-to-date の 2 つの関連で結ばれる．システム利用者が明示的にプロダクトにバージョンを付けた場合は history で関連付けられる．up-to-date は常に最新の Document オブジェクトが 1 つだけ Product と関連付けられる．

このオブジェクト図が示すように 1 つの Product オブジェクトが複数の Document オブジェクトを持つこと，即ちモデルに対して複数のドキュメント情報を持つことにより，1 つのプロダクトがバージョン毎に持つ多面性を表現している．

3.3.2 開発プロセス支援

モデリング環境上で起動されるモデラにおける開発プロセスをガイドすることで，モデル作成プロセスのナビゲートをサポートする．開発プロセスのガイドは，各モデラに対応した開発プロセスに関するガイド情報を持つガイドモデルをリポジトリサーバの Guide オブジェクトが解釈してシステム利用者に情報を提供することで行う．Guide オブジェクトは開発プロセスのガイド情報を，Web ブラウザを通じてシステム利用者に提供する．

ガイドモデルは，“ガイド” というドメインに特化して定義されたメタモデル（ガイドメタモデル）が持つモデラ（ガイドモデラ）を用いて作成する．ガイドメタモデルとガイドモデラはリポジトリシステムが最初から用意し，システム利用者はモデリング環境上でガイドモデルを作成できる．ガイドメタモデルを図 7 に示す．

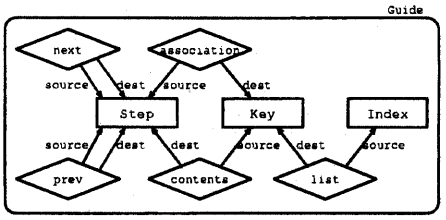

図 7: ガイドメタモデル

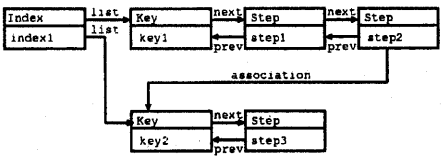

図 8: ガイドモデルの例

このメタモデルを構成するエンティティは，システム利用者に提供するガイド情報の種類を分類する Key，ガイドする開発プロセスの作業手順の各工程の情報を持つ Step，ガイドモデルでサポートしているガイドの種類を統合して持つ Index がある．これらのエンティティを関連付けるリレーションシップとなる要素は，next, prev, association, contents, list である．next, prev でエンティティ Step 同士を関連付けることで，Step で分割されている作業手順が順序付けられる．association はエンティティ Step と Key を関連付け，Step の作業内容で関連した項目との関係を表している．ガイドモデル内にあるガイド項目 Key は全て Index と list で関連付けられ，作業手順の最初の工程に対応する Step は Key と contents で関連付けられる．

このメタモデルで作成されたガイドモデルの例

を図 8 に示す．このメタモデルによってガイドモデルは，モデラ上での目的別に分類された作業手順のガイド，用語の定義や作業に関連した項目への移動といったものを表現することができる．ガイド情報は Web ブラウザを使って提供される．このため，これらの情報はハイパーテキストを利用して各項目がリンクで関連付けられて表示される．

3.4 データベースの操作

データベースに対する操作はリポジトリサーバの Database オブジェクトが行う．Database オブジェクトがデータベースに対して行える操作は connect, disconnect, store, restore, delete の 5 つで，それぞれデータベースへの接続，接続を確立していたデータベースの切断，プロダクトの格納，プロダクトの取り出し，プロダクトの削除に対応する．

4 分散環境への対応

リポジトリシステムの構成要素である，モデリング環境とリポジトリサーバは，必ずしも同じ計算機上で実行する必要はなく，それぞれの適した環境で動作することができる．このため，モデリング環境とリポジトリシステムは互いに分散環境に対応する必要があり，相互の通信に CORBA [7] を利用する．

4.1 Bramble の拡張

本モデリング環境やそのプロダクトは Bramble で記述されるため，Bramble に CORBA のインタフェースを持たせることによって，分散環境にも柔軟に対応できるようにした．Bramble に以下の三種類のオブジェクトを追加したことで，簡単に CORBA を利用できるようにした．

BOA（Basic Object Adapter）オブジェクトは，オブジェクトを外部からアクセス可能にするための処理や，外部からのリクエストの Bramble オブジェクトへのマッピングを行う．ORB（Object Request Broker）オブジェクトは，初期オブジェクトリファレンスを取得したり，オブジェクトリファレンスと文字列（IOR）の相互変換を行うためのインタフェースを持つオブジェクトである．OR（Object Reference）オブジェクトは，自身に送られたメッセージを OR が参照する（CORBA インタフェースを持つ）オブジェクトへ送信する．

また，外部の CORBA 対応オブジェクトを OR オブジェクトとして参照することによって，他の Bramble オブジェクトと同じようにメッセージ送信を記述することができる．

4.2 データベース制御

データベースの制御に関する操作は，Web ブラウザとモデリング環境から利用可能となる必要がある．そこでリポジトリサーバのデータベースの制御を持つ Database オブジェクトに CORBA を利用する．Database オブジェクトのこれらの操作を CORBA インタフェースとして定義し，モデリング環境からのオペレーション呼出しに応答する．これによってモデリング環境からは，Database オブジェクトがモデリング環境上にあるオブジェクトと同等に扱うことができる．

5 関連研究との比較

モデリング環境とリポジトリシステムそれぞれについて関連研究と比較する．

5.1 モデリング環境

本研究と同様メタ階層を用いるものとして，CDIF[5] があるが，CDIF は CASE 間のデータ変換を目的として定義されたものであるのに対し，本研究のメタ階層ではモデリングを支援するため，メタモデルに CASE ツール（モデラ）自体の定義が含まれモデルの表示的側面もモデラによって定義される．また，CDIF では，モデルの表示的側面を NA モデルで，意味的側面を ER モデルで表現するのに対して，本研究では，それぞれメタモデルの定義に任せられているので，モデルを表現できる領域が広いと言える．

また，ソフトウェア部品によってシステムを開発できる統合的なソフトウェア開発環境として，VisualBasic や VisualAge などが挙げられるが，本環境では，ソフトウェア（モデル）を定義するメタモデルを定義し，そのメタモデルを部品として新たなメタモデルを開発できるようにすることで，

よりドメイン特有なモデリング環境を提供することが可能となっている.

5.2 リポジトリシステム

まずプロダクトモデルの点で, 情報資源辞書システム IRDS[8] と本研究を比較する. IRDS はデータベースのデータ構造も含めた一般的な情報資源に対する構造を定義して格納するデータベースで, 4 層のメタ階層によってメタ情報を各階層で定義している. 従って格納される成果物のプロダクトモデルは, メタ階層の延長上に定義されていることになる. 一方, 本リポジトリシステムは, メタ階層に基づくモデリング環境から生成された成果物を管理対象としているが, 管理するプロダクトモデルはこのメタ階層の延長上にはない. 管理対象となる本モデリング環境の成果物は全てモデルになりそれ以外の成果物はないため, プロダクトモデル自身の定義を編集する必要はない. このことからプロダクトモデルを簡素に体系化することができた.

次に CASE ツールの統合の点で PCTE[9] と比較する. PCTE は, トースタモデルを使って CASE ツールから PCTE で標準化されたツールの各サービスのインタフェースを通じてリポジトリにアクセスすることで, 仕様の異なる様々な CASE ツールを統合している. 一方, 本リポジトリシステムにおける CASE ツールはモデラであり, これは各メタモデルが持つためドメイン毎にモデラが存在することになる. 従って, あるドメインのモデルを別のドメインのモデラで編集する必要はないため, これらのモデラを統合するようなことはない.

6 おわりに

モデルとそれを記述するメタモデルを分割し, その関係を規定した. そして, メタメタモデルというメタモデルを記述するためのメタモデルを定義することによって, モデル開発のレベルを分割し, ドメインに特化したモデリングを支援する環境を構築した. また, ドメインに特化した既存のモデルコンポーネントを組み合わせ, 新たなモデルを作成することを可能にしたことにより, エンドユーザコンピューティング環境が実現できたと言える.

更に, プロダクトのドキュメント化と開発プロセスの支援をするリポジトリシステムを構築した. リポジトリシステムはモデリング環境を対象としているため, 管理する成果物をモデルに統一することができ, その結果, 管理体系を簡素にすることができた. 更に開発プロセスを支援するガイドモデルをメタ階層に基づいて設計したことで, ガイドモデルはモデリング環境上で編集可能となり, システム利用者がモデラにガイド機能を容易に組み込むことができる.

またモデリング環境とリポジトリサーバ間に CORBA を利用することで, アプリケーションレベルでの柔軟な分散環境を可能とした.

今後, モデリング環境については, メタレベルモデルでの統合を考慮したメタモデル作成の支援環境の整備, 多くのドメインを調査し, それらをカバーできる一般的なコンポーネントの設計を進める. また, リポジトリシステムについては, 多人数利用を考慮した分担作業をサポートした開発プロセスの管理, データベース内のプロダクトに対するアクセス制御の導入を検討していく予定である.

参考文献

[1] 高橋 大輔, 上田 賀一: "メタ階層に基づくモデル構築とその支援環境", 近代科学社, ソフトウェア工学の基礎 III, 日本ソフトウェア科学会 FOSE'96, 146–149 (1996)

[2] 上田 賀一, 中野 喜之, 金村 星吉, 高橋 大輔: "オブジェクト指向モデル記述言語 Bramble の開発", 情報処理学会, 研究報告 (SE), Vol.96, No.32, pp.65–72 (1996)

[3] 上田 賀一, 石川 裕喜, 中野 喜之: "オブジェクト指向言語 Bramble による視覚的開発環境の構築", 情報処理学会, 研究報告 (SE), Vol.97, No.25, pp.41–48 (1997)

[4] フィル・サリー著, 本位田 真一 監訳: "オブジェクト指向モデリング", 日経 BP 出版センター (1995)

[5] 篠木 裕二, 西尾 高典, 吉川 彰弘: "CDIF—CASE データ変換形式", コンピュータソフトウェア, Vol.10, No.2,pp.13–25 (1993)

[6] 落水 浩一朗: "ソフトウェア・レポジトリ", 情報処理, Vol.35, No.2, pp.140–149 (1994)

[7] 小野沢 博文: "分散オブジェクト指向記述 CORBA", ソフト・リサーチ・センター (1996)

[8] 堀内 一: "情報資源辞書システム IRDS の標準化", 情報処理, Vol.37, No.7, pp.638–645 (1996)

[9] Lois Wakeman & Jonathan Jowett 著, PIMB 協会 編, 松本 吉弘 監訳: "PCTE: 開放型リポジトリのための標準 —ソフトウェア利用環境の基盤—", 日科技連 (1994)