ソフトウェア工学 140－5
（２００３．３．６）

メタ階層を用いたプロトタイピング手法の提案

大塚 聖也　　小飼 敬　　上田 賀一

茨城大学 工学部 情報工学科

〒 316-8511 茨城県日立市中成沢町 4-12-1

本研究では, 効率の良いプロトタイピング手法モデルとその手法に基づく支援環境を提案する. 本手法モデルでは, 開発フェーズを対象ドメインの基本モデルであるドメインモデル開発と対象システムのプロトタイプであるプロトタイプモデル開発に分割した. またドメインモデルの開発にはメタ階層概念を導入することで, 多様なドメインのモデルを統合的に扱えるようにした. さらにメタモデルの概形をメタメタモデルとベースモデルから生成させた. これらは, ドメイン特化な図式記述を可能にし, また支援環境側にて開発者の分離を可能にした. このことはユーザに求められる専門的知識や技術の分別も可能とする.

A Prototyping Approach Model Based on the Meta Hierarchy

Masaya OHTSUKA　　Kei KOGAI　　Yoshikazu UEDA

Ibaraki University

4-12-1 Naka-Narusawa, Hitachi, Ibaraki, 316-8511 Japan

This paper proposes a prototyping approach model and its modeling environment system with flexibly representing diagrams. This prototyping approach model divides software prototype development into two stages of domain modeling and prototype modeling. This approach adopts the meta hierarchy architecture. In domain modeling, developer makes a base-model with the smallest set of model elements at first, then developer draws out meta-model from this base-model under the support of modeling environment system. This environment leads user to make domain specific diagrammatic representation of target software system and distinguishes between the high and low-level users of special knowledge and technique.

1　はじめに

近年のソフトウェアシステムに対する要求は大規模化・複雑化の一途を辿っている. 背景にはハードウェア性能の向上により, ユーザの高度な要求受け入れが可能になったことや, ソフトウェアの適用分野の拡大などがある.

これらの状況下において, 対象物を簡単化・抽象化することで関係者相互のコミュニケーションを助長する実体関連図やデータフロー図など図式表現によるモデル化は, その有効性を認められ様々なソフトウェア開発に使用されている.

こういった CASE ツールの中には, 仕様記述言語を用いることによりシステムのプロトタイプ作成を支援するものがある. しかしそれらの多くが記述様式にデータフロー図や状態遷移図などと図式表現が固定されており, その記述言語の適用性が高いシステムでは有効であるが, 適用性が低い

システムでは有効ではない.

本報告では適用する問題ドメインに特化した図式表現を可能とすべく, 図式表現を固定しないモデリング環境を提案する. その際, プロトタイピング手法を用いたソフトウェア開発において, ドメインモデル開発とプロトタイプモデル開発を分割する.

ドメインモデル開発ではドメイン特化した図式表現とその制約や振舞いであるメタモデルを作成する. その際, メタメタモデルとベースモデルからメタモデルを生成する. プロトタイプモデル開発では, モデルの最小例であるドメインモデルをコピー・編集することでプロトタイプモデルを構築する.

このようにドメインモデル開発とプロトタイプモデル開発を分割することによりドメインモデル開発者とプロトタイプモデル開発者を明確に分離することが可能である. ドメインモデル開発には詳細な意味記述が必要であるので, メタ概念やプログラムに関する専門性が問われるが, プロトタイプモデルではそれらの専門性は薄れるので, 要求者を含めた迅速なプロトタイピングが望める手法である.

始めに本研究にてベースとしている ERF モデルについて述べ, その中でメタ階層アーキテクチャ [1][2][3] について述べる. 次に本研究にて提案するプロトタイピング手法と一般的なプロトタイピング手法について述べる. その後, 本支援環境の設計について説明し, 関連研究との比較を行ない, 最後にまとめと今後の課題について述べる.

2 ERF モデルとメタ階層アーキテクチャ

本プロトタイピング手法モデルでは, 構築したプロトタイプモデルを実際のソフトウェアの動作に近づけるためプロトタイプモデルを構築している図式表現にその振舞いを持たせることが可能である. 本手法モデルでは, メタ階層の最も抽象的なモデルとしてメタメタモデルに ERF モデルを採用している. 本節では ERF モデルとメタ階層アーキテクチャについて述べる.

2.1 ERF モデル

ERF モデルとは, 実世界を実体 (エンティティ) と関連 (リレーションシップ) で表す実体関連モデルを基本として, オブジェクト指向の有用性から実体と関連をオブジェクトとして扱うこととし, さらにオブジェクトの集約を扱う為に, 場 (フィールド) の概念を取り入れたものである.

- **エンティティ**
  モデリングの実世界を構成する実体オブジェクト
- **リレーションシップ**
  2 つのエンティティ間に存在する関連
- **フィールド**
  エンティティ, リレーションシップ, フィールドを集約するオブジェクト

これら ERF モデルの特徴としては以下のようなものが上げられる.

- 各要素をオブジェクトとしたので属性を定義可能
  先にも述べた様にプロトタイピングにおけるオブジェクト指向の有用性より, 各要素はオブジェクトとして扱う. そのため各要素は属性やメソッドを持つことが可能である.
- 関連の多重度を定義可能
  リレーションシップやフィールドに制約記述のための属性を設けることにより, 制約のチェックを可能とした.
- リレーションシップは 2 項関連のみ定義可能
  リレーションシップは 2 項関連のみを定義可能とした. これは 3 項関連以上は複数の 2 項関連にて記述可能であるということと, 制約の簡略化のためである.
- フィールドを用いることによりモデルの階層的な記述が可能
  フィールドを単位としてモデルを記述することが可能である. またフィールドはオブジェクトの場として用いることもできる.

2.2 メタ階層アーキテクチャ

本報告では, 要求者や開発者にとって対象ドメインを理解しやすい図式表現を使用して, プロトタイピングを行う際の手法モデルと支援環境の開発を目的としている. 図式表現とは, 実世界のモデル化であり, モデルはその目的や対象ドメインによって様々な種類が存在する. 例えば, データフローモデル, 状態遷移モデル, 実体関連モデルなど

があり, これらはより一般的な目的のために使用されるモデルの種類である. また, よりドメインに特化したモデルとして, 待ち行列モデルやフィルタモデルといったモデルの種類が考えられる. 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対象ドメインでの記述効率が高いと言える. そこで, このような様々なドメインのモデルを統合的に扱うことができるよう, 「メタモデルはモデルを記述・解釈する」と定義した.

メタ階層アーキテクチャでは, モデルを記述・解釈するメタモデルと同様に, メタモデルを記述・解釈するメタメタモデルを定義する. その際, メタメタモデルを記述・解釈するメタメタメタモデルを定義することも可能であるが, ドメインにおけるモデルとメタモデル, さらにメタモデルとメタメタモデルの関係によりメタメタモデルまでの定義で十分であると判断し, メタ階層アーキテクチャにおける最も抽象モデルとしてはメタメタモデルと定義した. また, メタモデルによって記述・解釈されるモデルをベースモデルと呼ぶこととする.

2.2.1 メタメタモデル

メタメタモデルはメタ階層アーキテクチャにおける最も抽象モデルであり, メタメタモデルに ERF モデルを採用している.

メタメタモデルの構成要素は図 1に示すように, ENTITY, RELATIONSHIP, FIELD の 3 つの実体と, これらの実体間の relate, contain の 5 つの関連である. メタメタモデルはこれら 8 つのコンポーネントからなる.

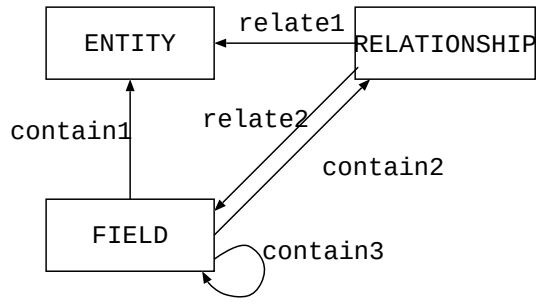

図 1: ERF モデルによるメタメタモデル

2.2.2 メタモデル

メタモデルのコンポーネントとその間の関連は, メタメタモデルの定義に基づきドメイン毎に記述される. その際, 各コンポーネントの属性やメソッドについても定義する.

本報告においてメタモデルによって記述・解釈されるベースモデルは, ドメイン特化した図式表現である. 図式表現は視覚情報と意味情報に分割が可能であり, さらに視覚情報, 意味情報ともにメタ階層を導入することが可能である.

今回, 要求者と開発者にとって対象ドメインが理解しやすく, 実際のソフトウェアの動作に近似したプロトタイプモデルを作成する図式表現を目的としているので, 視覚情報のメタモデルの編集可能範囲が広大であると, 意図とは逆に理解しづらい図式表現となってしまう可能性もある. また, 視覚情報と意味情報を分割することはより柔軟なモデル定義が可能になる一方で, 記述量の増大と複雑さをまねく. よって本報告では, モデルは視覚情報と意味情報に明確に分割せず, また, 視覚情報のメタモデルに制限をかける. 円, 矩形, 角丸矩形, 多角形といった視覚情報における図式要素は Entity または Field をメタメタモデルとして持つとし, 直線は Relationship をメタメタモデルに持つとする. さらに, メタモデルにはモデルの意味的なメタモデル, つまり, 制約や振舞いも記述するものとする.

2.2.3 ベースモデル

ベースモデルはシステムのプロトタイプであり, メタモデルに基づいて記述される. その際, メタモデルで記述されたコンポーネント間の制約や, フィールドが内部に持つことが可能なコンポーネントの制約を満たさなければならない.

要求者を含んでの迅速なプロトタイピングを目指すことから, 要求者にも理解容易となるようなドメイン特化した図式表現を, 本報告ではベースモデルとしている. また, ベースモデルをドメインモデルとプロトタイプモデルに分割している. ドメインモデルではシステムを構成する最小例を作成する. プロトタイプモデルはドメインモデルをコピー・編集することで目的のプトロタイプを構築する. この様に分割することで, 各モデルの開発

者を分割することが可能になり，それは開発者に必要とされる対象ドメインやプログラミング，メタ階層に関する専門性を見直すことが可能となる．

3　プロトタイピング手法

3.1　本研究提案のプロトタイピング手法

一般的なプロトタイピング手法によって生成されたソフトウェアのプロトタイプは，単に要求者と開発者間の思考上の差異を埋めるだけに使用されるものと，開発者によってさらに精錬されて要求者に納入されるものに大別される．本手法が目指すプロトタイプは前者である．その際，要求者や開発者に理解容易となるような図式表現を使用してプロトタイプを構築し，またソフトウェア完成時に近似した動作をすることで要求者からより多くのフィードバックを得ることを目的としている．さらに，プロトタイプの構築を，ドメインモデルの作成とプロトタイプモデルの構築との 2 段階に分割する．これにより開発者を，ドメインモデル開発者とプロトタイプモデル開発者へ明確に分離可能となる．図 2に，本プロトタイピングモデルを示す．

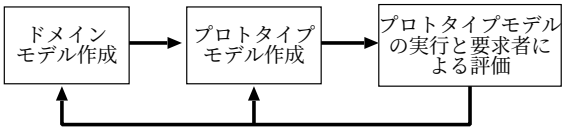

図 2: 本プロトタイピングモデル

3.1.1　ドメインモデルの作成

ドメインモデルの作成はドメインモデル開発者によって行われる．ドメインモデルは，メタ階層におけるベースモデルとメタモデルの組である．ドメインモデルにおけるベースモデルとは，要求者や開発者が理解容易となる図式表現であり，プロトタイプを構築する際の最小例である．またメタモデルは，各図式表現に持たせる制約やプロトタイプモデル完成時に実行させる振舞いである．

メタ階層を導入した際のモデル開発手順としては，抽象モデルであるメタモデルを作成してから具象モデルであるベースモデルを作成する流れが一般的である．しかしゼロからメタモデルの作成を行うのはベースモデルの作成と比較して困難であるため，本手法では，ベースモデルを作成した後メタメタモデルとベースモデルからメタモデルを作成する手順を採用する．その際，本研究にて提案する支援環境はメタモデルの概形を生成し，ドメインモデル開発者がメタモデルの確認・編集を行いメタモデルを完成させる．またメタモデルは ERF モデル言語にて記述される．

これらの理由により，ドメインモデル開発者は，対象ドメインとメタ階層，ERF モデル言語についての高い専門性を必要とする．

図 3にドメインモデル作成の例を示す．

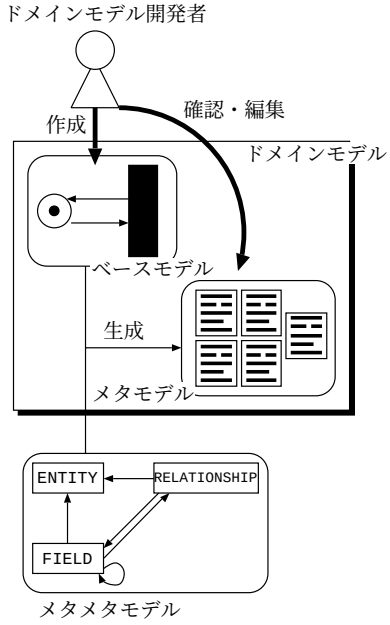

図 3: ドメインモデル作成の例

作成されたドメインモデルは目的のソフトウェアに使用されるだけではなく，同様のドメインが対象となった異なるソフトウェアにも，適用可能である．ドメインモデルの再利用によって，さらに迅速なプロトタイプの開発が望める．

3.1.2　プロトタイプモデルの作成

プロトタイプモデルの作成はプロトタイプモデル開発者によって行われる．プロトタイプモデル

は, メタ階層におけるベースモデルであるドメインモデルをコピー・編集することにより構築される. その際, プロトタイプモデル開発者はドメインモデルにて作成された図式表現のみを使用してプロトタイプモデルの構築を行うので, プロトタイプモデル開発者は, 各図式表現のメタモデルや対象ドメインの全ての要素がモデル化されているかといったことの確認作業をせずともよく, プロトタイプの構築に集中できる. このことから, プロトタイプモデル開発者は ERF モデル言語や対象ドメインの高い専門性をもっていなくとも, 要求者によって依頼されたソフトウェアのプロトタイプの構築が望める.

図 4にプロトタイプモデル構築の例を示す.

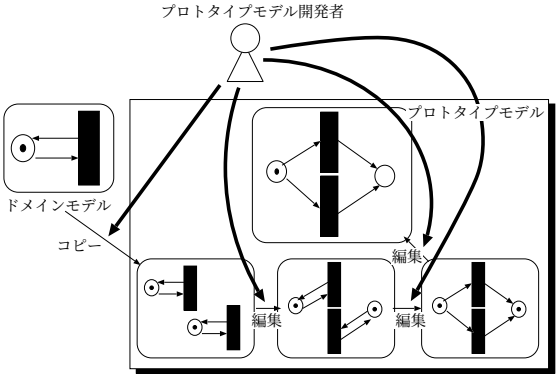

図 4: プロトタイプモデル構築の例

3.1.3 プロトタイプモデルの実行

本手法でのプロトタイプは, プロトタイプを反復的に精錬することで最終的に要求者に納入されるソフトウェアを作成する目的ではない. 要求者と開発者間の思考上の差異を埋めることを目的とした分析用の道具として使用することを目的としている. 一般的な分析目的のプロトタイプでは, ソフトウェアの実際の動作はわからないことが多い. 本手法でのプロトタイプは, 実際のソフトウェアに近似した動作を実行することが可能である.

プロトタイプモデルの動作や制限は, ドメインモデル開発時に作成されるメタモデルで記述される. その際, メタモデルは ERF モデル言語で記述される. ERF モデル言語は, 当研究室にて研究されているモデル記述言語である. 本研究でも用いている ERF モデルを基本要素として考慮したオブジェクトベース言語である.

プロトタイプモデル作成フェーズに移行した後は, いつでもその時点でのプロトタイプモデルの実行が可能である. 要求者から実行結果を確認してのフィードバックを受けプロトタイプモデルを編集するため, プロトタイプモデルには完了時点を明確にすることが困難なである. よって, いつでも実行を開始できる.

また, 本支援環境によって提供される ERF モデル解釈機構では, 実行中のプロトタイプモデルを任意の時点で中断し, モデルのパラメータを編集し再開可能を目指す. これにより, より柔軟なプロトタイピングが可能である.

図 5に, プロトタイプモデルの実行の概念を示す.

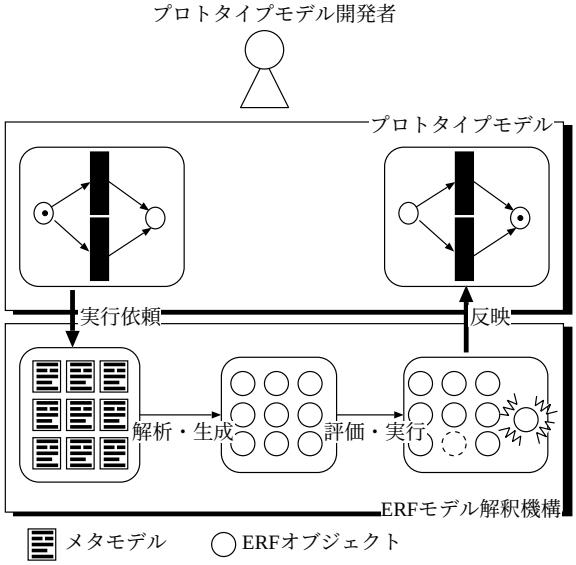

図 5: プロトタイプモデルの実行概念

4 本支援環境

本支援環境は, 前述のプロトタイピング手法モデルのサポートを目的としている. その際, オープンソースソフトウェアの Argo/UML[7] を基に ERF モデルの概念と明確化した 2 つの開発フェーズ間の移行をサポートしている. 本節では, 支援環境の機能と設計を述べ, 使用例を示す.

4.1 本支援環境の機能

本支援環境は, 前述のプロトタイピング手法モデルをサポートするため次の機能を有し, 各モデル開発者に提供する. 図 6に本支援環境の概念を示す.

- ドメインモデル開発とプロトタイプモデル開発という 2 つの開発フェーズ間の移行
  開発フェーズの移行をサポートすることにより, モデル開発者間の明確な移行が可能である. このことから, 各モデル開発者に適した機能を提供できる.
- ドメインモデル開発時に, 作成されたベースモデルと環境側で提供するメタメタモデルからメタモデルの概形を生成
  メタモデルをゼロから作成することは, ベースモデルの作成と比較して困難である. そこで本支援環境では, ベースモデルとメタメタモデルからメタモデルの概形を生成する. 本支援環境では, メタモデルは ERF モデル言語によって記述される. メタモデルにはベースモデルの制約や振舞いが記述され, 制約は作成された図式表現から環境によっての概形生成が可能である. 一方, 振舞いは生成が困難であるので, ドメインモデル開発者によって直接記述する必要がある.
- GUI による図式表現作成・編集, プロトタイプモデルの構築
  本支援環境が基としている Argo/UML は GUI を使用しての UML 作成を目的とした CASE ツールである. Argo/UML が持つ GUI を使用してのモデル作成機能を取り込み, UML のみ使用可能であった図式表現を, 図式表現の編集が可能にした. また, ドメインモデルをコピー・編集してプロトタイピングモデルの構築が行えるよう拡張した.
- プロトタイプモデルの評価・実行
  プロトタイプモデルの実行時は, その制約, 振舞いを記述しているドメインモデルのメタモデルを評価・実行する. メタモデルの記述を ERF モデル言語解釈機構が解析して, ERF オブジェクトを生成し, 実行される振舞いを評価する. ERF オブジェクトの変化はプロトタイプモデルに反映される.
- メタメタモデルである ERF モデルを概念としてサポート
  メタ階層における最も抽象的モデルとしてメタメタモデルを定義し, ERF モデルを採用している. これはメタメタモデルによって記述されるメタモデルに影響し, メタモデルには ERF モデル言語を採用している.
- 作成されたドメインモデルやプロトタイプモデルの保存と読み出し
  ドメインモデルとプロトタイプモデルは作成された組での使用だけではなく, 新たなプロトタイプモデルの構築時に既存のドメインモデルを再利用可能である. それは対象ドメインが同じで要求されたソフトウェアが異なる場合などである. この場合, ドメインモデルをゼロから作成する必要はなく, より効率的なプロトタイピングが望める.

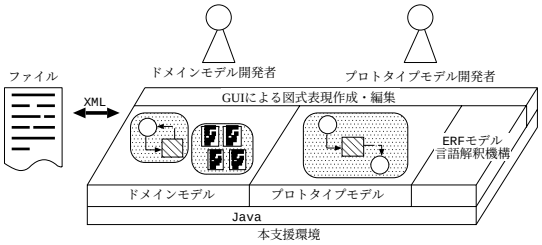

図 6: 本支援環境の概念図

4.1.1 グラフモデル作成機能

本支援環境は Argo/UML のモデル作成機能をベースに UML 以外の図式表現の作成, 使用を可能に拡張している. 図式表現を作成する際には基本的な図式要素である, 円, 矩形, 角丸の矩形, 多角形, 直線を使用する. 本手法モデルではメタ階層を導入し, メタメタモデルである ERF モデルを考慮に入れると, メタモデルは実体, 関連, 場のいずれかに分類される. 円, 矩形, 角丸の矩形, 多角形はメタモデルとして実体もしくは場として分類され, 直線は関連として分類される.

円, 矩形, 角丸の矩形, 多角形は実体もしくは場であるから単独での作成と存在が可能である. しかし直線は関連であるので, 作成時には関連元と関連先が必要となり, また, 関連元もしくは関連先が消去された場合, 関連自身も消去される. 円, 矩形, 角丸の矩形, 多角形の内側に他の要素が配置された場合, 外側の要素は場に分類される. 場の内側に配置された要素は, 場の内側のみで生成され、場が消去された場合, 要素も消去される.

本手法モデルではドメインモデル開発とプロトタイプモデル開発に開発フェーズと開発者を分割している. 基本的な図式要素を使用するのはドメインモデル開発時のみであり, プロトタイプモデル開発時には使用しない. その代わりプロトタイプモデル開発時には, ドメインモデルを一括でコピーする機能と, コピーされたモデル中に存在する関連の関連元や関連先の編集機能がある.

4.1.2 ERF モデル言語解釈機構

ドメインモデル開発時に作成されたメタモデルとプロトタイプモデルの構築過程は ERF モデル言語によって記述される．プロトタイプモデルの構築ファイルには，ドメインモデルの全メタモデルとベースモデル，プロトタイプモデルの構築手順が記述され，構築ファイル単体で解釈・実行が可能である．また，ERF モデル言語解釈機構は Java にて実装される予定である．

ERF モデル言語解釈機構は主として次の機能を持っている．

- モデル構築ファイルの ERF モデル言語を解析
  ERF モデル言語仕様により，記述ファイルを解析する．
- 解析結果による ERF オブジェクトの生成
  ERF オブジェクトも Java にて実装される．解析結果によって ERF オブジェクトの生成・属性の変更を行う．
- 生成された ERF オブジェクトの実行とモデルへの反映
  実行依頼を受けた振舞いを評価・実行し，その結果 ERF オブジェクトへの変化があった場合，それをプロトタイプモデルへと反映させる．

5 関連研究

本論文にて述べたプロトタイピング手法モデルとその支援環境について，関連研究と比較・検討する．

5.1 UML Virtual Machine

UML Virtual Machine(以下，UML 仮想機械) [5] は，モデリング時に多く使用される UML 記法を評価させる仮想機械である．主な特徴としては次のようなものが挙げられる．

- アーキテクチャの分割
  UML 仮想機械のアーキテクチャは，論理アーキテクチャと物理アーキテクチャに分割される．論理アーキテクチャはオブジェクト間の関係についての論理構造を記述し，物理アーキテクチャは論理アーキテクチャの実現部分である．
- メタ階層の導入
  図 7に示すように，UML 記法は 4 レベルの構造である．M3 はメタメタモデルレベルであり，UML によって使用される言語を定義する．M2 はメタモデルレベルであり，ここが UML 記法にあたる．M1 はモデルレベルである．ユーザ定義のクラスやドメインによるクラスに相当する．M0 はユーザオブジェクトやドメインオブジェクトのレベルである．UML 仮想機械はこの 4 階層をサポートする．
- UML 仮想機械は Java にて実装
- モデルの実行
  UML 記法は，対象モデルの分析・設計に主眼があり，振舞いを明確に記述することは定義してない．そこで，UML 仮想機械では，UML 記法における状態遷移図をベースとして，詳細な振舞い記述に OCL(Object Constraint Languages) を採用している．OCL は UML のバージョン 1．1 から導入され UML の意味記述に使用されている．

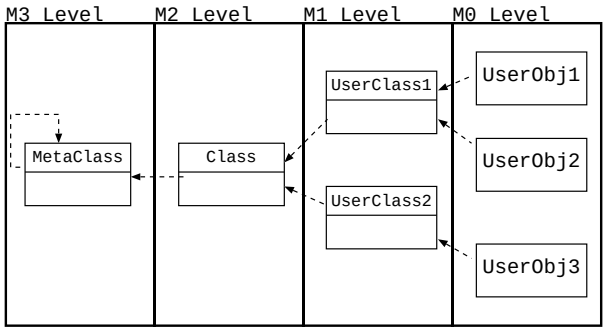

図 7: UML における 4 階層

5.1.1 本報告との比較

UML 仮想機械は図式表現に UML 記法のみをサポートしているのに対し，本研究では図式表現の非固定化を行った．結果として UML 仮想機械では，どのようなドメインに対しても UML でモデリングを行い，本研究ではそのドメインに特化した図式表現でのモデリングを行う．UML 記法は決定している仕様の設計には有効であるが，要求者を含んでのプロトタイピングのような流動的な仕様の設計では有効性が薄い．また，UML 記法によるモデルを実行させ確認した場合，確実に理解しているのは開発者だけで要求者は理解していない場合が考えられる．

UML 仮想機械は開発者の分離は行われておらず，開発者は UML と OCL，対象ドメインに関する専門性を持っていなければならない．本研究で

は, 開発フェーズを分離することにより開発者の分離を行い, 各開発者に必要な専門性の見直しを図っている.

これらから, 要求者を含んでのプロトタイピングの際には, 本研究にて提案したドメインに特化した図式表現を許可する環境が望ましいと言える.

5.2 高橋らによる研究

当研究室の高橋らによる研究 [2] では, ユーザによるメタモデル作成を含むモデリング支援環境の提案がなされた.

高橋らによる研究の特徴は次の通りである.

- GUI によるモデリング支援とモデルにメタ階層を導入
  メタメタモデルには ERF モデルを採用して, メタモデルの記述はモデルと同じく GUI による図式表現にて行う.
- 開発者をメタモデル開発者とモデル開発者に分離
  メタモデル開発者によってメタモデルの作成が終了した後, モデル開発者によってモデルが作成される.
- Bramble にて支援環境を実装
  オブジェクト指向モデル記述言語 Bramble[3] にて支援環境を実装した.

5.2.1 本報告との比較

高橋らの研究におけるメタモデル開発者は, 本研究におけるドメインモデル開発者と同じ位置付けである. 共に, 次の開発フェーズにて使用される図式表現とその制約や振舞いを作成する. 高橋らの研究におけるメタモデル開発者は, メタモデルを作成する場合ゼロから作成しその後図式表現を作成する. 本研究でのドメインモデル開発者は, 図式表現を作成してからメタモデルを作成し, さらにメタモデルは支援環境にてその概形が生成される. このことから, 本研究にて提案する手法モデルの方が, メタモデル開発の容易性・効率性が良いと思われる.

また高橋らの研究では, メタモデルの作成時にモデルにて使用される図式表現のイメージを既に持っているにもかかわらず, メタモデルから作成しなければならない. 本研究のアプローチは図式表現を作成した後にメタモデルを作成するので, より自然にモデル開発が可能である.

6 まとめと課題

本報告では, 開発フェーズをドメインモデルの開発とプロトタイプモデルの開発に明確に分離することで, 開発者もドメインモデル開発者とプロトタイプモデル開発者に分離することが可能となった. また, ドメインモデル開発時に多様なドメインでのモデルの記述を容易かつ効率的に行うためメタ階層アーキテクチャを導入し, メタメタモデルに ERF モデルを採用した. またベースモデルとメタメタモデルからメタモデルをの概形を生成する支援環境を設計した.

課題として, ERF モデル言語解釈機構の完成と, 本支援環境では考慮していない要求分析による入力データの拡張が挙げられる.

参考文献

[1] 安間その美 : “モデルのメタ階層に基づいたソフトウェアプロトタイピング基盤機構の開発”, 情報処理学会, 第 52 回全国大会, No. 5, pp. 65-66(1996, 3)

[2] 高橋大輔 : “メタアーキテクチャに基づいたモデル構築支援環境の開発”, ソフトウェア工学の基礎 III, pp. 146-149, 近代科学社 (1996, 12)

[3] 中野喜之 : “メタ階層アーキテクチャに基づくモデリング環境の設計と実現”, ソフトウェア工学の基礎 IV, pp. 71-74, 近代科学社 (1997, 12)

[4] 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)

[5] Dirk Riehle, Steven Fraleigh, Dirk Bucka-Lassen, Nosa Omorogbe: “The Architecture of a UML Virtual Machine”, OOPSLA ’01. ACM Press, 2001

[6] 塚本 明, 小飼 敬, 上田 賀一: “要求モデルとドメインモデルの関係に基づく分析支援環境の構築” 茨城大学大学院理工学研究科情報工学専攻修士論文 (2003, 2)

[7] Argo/UML プロジェクトホームページ: http://argouml.tigris.org/