TAMEデータモデル：
ソフトウェア経験ベースのための
オブジェクト指向データモデルの設計

野呂昌満，Parke Godfrey，Victor R. Basili

TAME: Data Model
Object Oriented Data Model for
Software Experience Base

南山経営研究
第9巻　第2号　抜刷
1994年11月

Nanzan Management Review
Vol. 9 No. 2 Offprint
November 1994

〈南山経営研究　第9巻　第2号　1994年11月〉

TAMEデータモデル：ソフトウェア経験ベースのためのオブジェクト指向データモデルの設計

野呂昌満，Parke Godfrey，Victor R. Basili

ソフトウェア開発管理・支援システムの経験ベースのためのデータモデルについて述べる。経験ベースはソフトウェア開発における経験を保持する知識ベースであり，ソフトウェア開発管理・支援システムの核となる。筆者らは，過去にソフトウェア開発を多数観察した経験に基づき，ソフトウェアプロジェクトの経験を知識として格納し利用するためのオブジェクト指向データモデルを設計した。このデータモデルは，従来のオブジェクト指向データモデルにあった機能に加えて，制約された型付け，ファセットに基づく多重is-a関係，オブジェクト属性の動的統一などの機能を規定する。

1．はじめに

筆者らは，過去10余年にわたって種々のソフトウェア開発プロジェクトを観察してきた結果，有効な開発工程，技術，道具などは応用分野ごとに異なるものであると考えるに至った[BAS88]。この背景にはソフトウェアの応用範囲が非常に多岐にわたるという事実がある。個々のソフトウェア開発上の諸問題には，それぞれに対してもっとも適した解決策が存在し，それらはプロジェクトごとに異なるものだと考えられる。

この前提を受け入れると，ソフトウェア開発にとって経験の再利用と蓄積が必要不可欠になる。ここで，

経験＝｛e｜eは経験則｝

経験則＝(p, c, r)

ここでp＝あるソフトウェア開発プロジェクト，

c＝（開発工程，技術，道具）：開発環境，

r＝プロジェクト実行結果

である。すなわち，あるプロジェクトに最適な開発環境は過去の同種のプロジェクトの経験に基づいて計画されるべきである。また，どのソフトウェア開発プロジェクトにおいても，将来の再利用に向けて，経験の改良（質と量の両面での向上，すなわち不要な経験則の削除と必要と認められた新たな経験則の追加）を行なうべきである。

そのために有効な方法として，個々のソフトウェア開発プロジェクトを実験として捉え，実験結果の考察に基づいて経験を改良するという自然科学的手法が考えられる。

筆者らはTAMEプロジェクト［BAS88］において，上で述べた，実験に基づく経験の改良をソフトウェアプロジェクトに応用し，その有効性を検証し，確認してきた。実験に基づく経験の改良を規定するパラダイムをTAME［BAS89］と呼ぶ。TAMEは品質改良パラダイム（Quality Improvement Paradigm）に目的/質問/尺度パラダイム（Goal/Question/Metric Paradigm，以下G/Q/Mパラダイムと呼ぶ）をその1つの段階に融合したものである。品質改良パラダイムは経験の改良のための枠組みを規定する方法論で，大きく分けて

- プロジェクト環境の特徴付け
- 実験と開発工程の計画，
- 計画の実行，
- 実験結果のフィードバック

の4段階からなる。G/Q/Mパラダイムは制御実験を計画するための方法論であり，ソフトウェアプロジェクトにおける実験を計画するために用いる。品質改良パラダイムの第2段階にG/Q/Mパラダイムを用いた方法論がTAMEである。

TAMEシステムはTAMEの各段階の計算機による自動化を目指して設計・実現しているソフトウェア開発・管理支援システムであり，経験ベース管理システムはその中核である。筆者らは過去にソフトウェアプロジェクトを観察したさいに得られたデータを経験ベースにどのように自然な形式で保存するかという問題に取り組み，その結果，経験ベースのデータモデルを得た。筆者らが設計したデータモデルは既存のオブジェクト指向データベース管理システム［BAN87，AND87，MAI86］のデータモデルの拡張であり，以下の3つの，従来にない特徴的な機能を規定する。

- 制約された型付け。
- ファセットに基づく多重is-a関係。
- オブジェクト属性の動的統一。

筆者らは，TAMEプロジェクトの一環として，データモデルをフレーム処理システムの拡張として実現し，ソフトウェアプロジェクトに応用する試みを行なっている。本論文では，次節でTAMEの説明をしたあと，データモデルの設計・実現について議論する。

2．TAMEプロジェクト

TAMEプロジェクトはパラダイムTAMEがソフトウェアの品質向上のために有効であるかどうかを検証することを目的とした研究プロジェクトである。これまで

に，手作業でTAMEをいくつかのソフトウェアプロジェクトに応用し，その有効性を実証してきた。本節では，以下の議論を円滑に進めるために，品質改良パラダイムとG/Q/Mパラダイムについて説明し，TAME実践のためにはプロジェクト実行組織と経験工場を分離する必要があることを示す。

2.1 品質改良パラダイム

品質改良パラダイムは以下の段階から成る：

(1) プロジェクト環境の特徴付け

(2) 実験と開発工程の計画，

(3) 計画（プロジェクト）の実行，

(4) 実験結果のフィードバック

第1段階（プロジェクト環境の特徴付け）では，当該プロジェクトにおける資源の使用，開発日程，ソフトウェアの大きさ，および開発環境に関するデータを与え，過去の同種のプロジェクトを同定する。過去のプロジェクトにおける経験は第2段階で再利用する。

第2段階（実験と開発工程の計画）では，当該プロジェクトで行なう実験を計画する。すなわち，実験の目的を決定し，収拾すべきデータとその解析および解釈の枠組みを定義する。つぎに，前段階で得られた過去の経験則に基づいて，当該プロジェクトにもっとも適した開発環境(開発工程，技術，道具の組)を選び出し，当該プロジェクト用にあつらえる。さらに，データ収拾の日程を考慮して，データ収拾作業を開発工程に折り込み，開発・管理工程を定義する。こうして定義した開発・管理工程をプロジェクト実行工程と呼ぶ。

第2段階で計画する実験は，つぎの2種類に大別できる：

(1) 経験改良指向実験

経験の改良を目的とする実験。たとえば，ある技術がある応用に対して有効であるかどうかを確認する等。

(2) 制御指向実験

プロジェクトを制御することを目的とする実験。たとえば，プロジェクトの進捗状況が計画どうりかどうかを確認する等。

これらの目的はたがいに相入れない場合があるので，この段階で，目的の優先順位を決定したり，別のプロジェクトを比較実験のために計画するなどして，矛盾を解消しなければならない。

第3段階（計画（プロジェクト）の実行）では，第2段階で得たプロジェクト実行工程に基づき製品を開発する。この段階では，実行工程に従って収拾したデータのうち，おもに制御指向実験に係わるものを解析・解釈し，プロジェクトに対して実時間

フィードバックを行なう。プロジェクト実行モデルには，実時間フィードバックに対する反応，代替案（Contingency Plan）を記述してあるので，それを実行する。

第4段階（実験結果のフィードバック）では，プロジェクト実行中に収拾したデータを解析・解釈し，実行結果を経験としてフィードバックする。この段階では，肯定的な結果はもちろん，否定的な結果も反例としてその理由を記した文書とともに経験ベースに保持する。

2.2 G/Q/M パラダイム

G/Q/M パラダイムは以下の段階からなる：

(1) 実験目的の設定

何（例えばある設計法）を何（例えば開発の日程を縮小するのに効果的かどうかを確かめる）のために実験するかをあらかじめ定められた型紙［BAS88］に従って定義する。

(2) モデルの定義

- 実験対象のモデル，
- 品質モデル：その実験で確認したい対象上の視点を示す，および
- フィードバックモデル：実験結果のフィードバックの筋書きを規定する

を定義する。ここでモデルを定義する理由は，実験の再現性（Tractability）を高めるためである。

(3) モデルと実験目的の結合

実験目的に従って，実験対象モデル，品質モデル，およびフィードバックモデルを関連づける。

(4) 目的の洗練化

目的を質問（下位の目的）に分解する。各質問の答が，データの組で定義できる水準まで分解を行なう。洗練化は容易ではないので，実験対象に応じた質問の組が経験則として得られており［BAS88, BAS89］，それを利用して分解を進める。

2.3 プロジェクト実行組織と経験工場

TAME は経験の改良を前提としたパラダイムなので，その実行のさいに，実験計画やデータ収拾作業などの費用を払わなければならない。これらの作業は，従来のソフトウェア開発においては必要がないものであり，短期的視点からすれば無駄である。しかし，経験を改良し再利用するという，長期的な視点からは，必要不可欠なものである。

上の議論の示唆するものは，TAME の実行は相反する利益を代表する異なった組

織が共同して行なうことが必要だという事実である。これらの組織の一方は時間内あるいは予算内に製品の完成を目指す組織であり，もう一方は，経験の改良を主目的とする組織である。筆者らは前者をプロジェクト実行組織（Project Organization），後者を経験工場（Experience Factory）と呼んで区別している。

プロジェクト実行組織の主要な興味は，通常のプロジェクトチームのそれと同じものである。したがって，プロジェクト実行組織は制御指向実験に興味を持つ。すなわち，実時間フィードバックによって，開発を円滑に行なうことを主要な目的とする。

一方，経験工場は経験改良指向実験に興味を抱き，プロジェクトが終了するごとに，事後フィードバックによって経験を改良する。また，経験工場は，品質改良パラダイムの第2段階で過去の経験をプロジェクト実行組織が再利用可能なように提供する。これらの過程で経験工場が行なっているのは，経験の分析と合成である。

3．TAME 経験ベースとその管理システムの設計

経験ベースは経験工場の中に存在し，ソフトウェアプロジェクトで得られた経験を保持する知識ベースである。上で述べたとおり，経験工場の行なうべき仕事は

- 再利用のための経験の提供，
- 経験の改良，および
- 経験改良指向実験の計画

である。経験の再利用のためには，経験則の構成要素，すなわち，開発工程，技術，道具や実験計画を抽象化したかたちで保持する必要がある。一方，経験則は，その定義からわかるように，これらの構成要素を関連づけたものである。これらを自然な形式で保持するためには，表現能力が高いデータモデルに基づく知識ベースとして経験ベースを設計しなければならない。

経験ベースのインターフェースを考え，経験ベース管理システム（経験ベースの検索・更新を処理するシステム）を実現すれば，経験工場で行なう仕事の自動化の第一歩は達成できる。経験の供給のためには，経験ベースを検索することが不可欠であり，経験の改良作業は管理システムを介して経験ベースを検索し，その一部を変更することで行なえる。さらに，実験の計画書も経験ベースに保持されるので，経験改良指向実験の計画にも経験ベースの検索は必要である。どのような検索機能が必要であるかについては，3.2節で議論する。さらなる自動化を考えた場合，不完全情報による検索，学習パターンを自動認識することによる経験改良の半自動化など，人工知能の研究成果を応用する必要がある。

経験ベースをこのように考えると，その設計目標は以下の2点である：

(1) データをモデル化する能力が高いこと。

(2) 人工知能ツール，言語処理系等がその上で実行可能な開放性(Open-endedness)を持つこと。

これらの目標を達成するように，われわれは経験ベースを，フレームシステムを拡張したオブジェクト指向データベースとして設計・実現した。

3.1 経験ベースに格納すべきデータ（モデル）

過去にソフトウェアプロジェクトを観察してきた結果，筆者らは，経験ベースに保持すべき経験則の構成要素として以下のモデルを定義した。モデルとはTAMEで扱う情報を，論理的に区別がつく単位に分割し，不要な詳細を省略したものである。

(1) プロジェクトモデル（Project Models）

TAMEの第一段階（プロジェクトの定義）で必要なデータ，すなわちプロジェクト環境を特徴づけるのに必要なデータをプロジェクトごとにまとめたもの。具体的には：

- プロジェクトの種類（事務データ処理ソフトウェア開発，基本ソフトウェア開発，等）
- 開発プロセス
- 使用計算機（対象計算機，開発用計算機）
- 開発言語（プログラミング言語，設計言語，要求仕様定義言語，等）
- プロジェクトの日程（schedules）や工程目標（milestones）
- プロジェクトの成否

などに関するデータをまとめたもの。

(2) 工程モデル（Process Models）

当該プロジェクトにおけるソフトウェア開発工程を記述したもの。開発工程の計画に用いる。

(3) 製品モデル（Product Models）

開発工程の結果として得られる最終製品であるソフトウェアや，その過程で得られる中間製品（設計仕様書，要求仕様書等）とそれらの記述項目や形式を規定したもの。開発工程の各段階の入力や出力を規定するために用いる。

(4) 資源モデル（Resource Models）

資源（ハードウェア，ソフトウェア，人，等）の使用や割当のためのモデル。実験計画（プロジェクト実行工程の定義）段階で資源の使用を予測・計画するために用いる。

(5) 品質モデル（Quality Models）

実験対象の確認すべき視点における品質の優劣を規定するモデル。例えば，使用者の観点から見たテスト技術の優劣を判断するためのモデル。実験結果の

解析および解釈に用いる。

(6) G/Q/Mモデル（実験計画書，G/Q/M Models）

実験の筋書きを記述したもの。実験対象とその品質モデルを関連づけ，フィードバックの筋書きを保持する。実験計画の結果得られる。

以上のモデル同士の関係として経験ベースに保持すべきものとして，同種のモデルを分類する一般－特殊関係，包括的なモデルを表現するための全体－部品関係，経験則を表現するための異種のモデル間参照関係が必要なことが，これまでプロジェクトを観察してきた経験から結論付けられる。

(1) 一般－特殊関係（General-specific relations）

この関係は既存のモデルを一部洗練化して新しいモデルを作成するときにそれらのモデル間に生ずる関係であり，モデルの分類を表す階層を形成する。

(2) 全体－部品関係

プロジェクトモデル，工程モデルなどその構成要素が同種または異種のモデルである場合，その包括的なモデルとその部品との関係を表す。

(3) 経験則を表現する関係

プロジェクトモデル，G/Q/Mモデルなど異種のモデルを関連づけて経験則として保持するさいに，それら異種のモデル間の関係をあらわす。

TAMEに基づいてソフトウェア開発・管理を支援するためには，以上に示したモデルとそれらの関係以外に，学習結果文書（Lessons Learned Documents）ならびに収拾したデータを経験ベースに保持する必要がある。学習結果文書は経験が記録された経緯や理由を記述したものなので，経験則を表現する関係と関連づけて記憶させる必要がある。経験則は，プロジェクトモデルまたはG/Q/Mモデルからその他のモデルへの関係で表現されるので，学習結果文書は，プロジェクトモデルまたはG/Q/Mモデルの部品として保持される。

TAMEにおけるデータ収集は計画され管理されたものなので，収集データは直接参照されるべきではない。つねにプロジェクトモデルまたはG/Q/Mモデルの部品として保持される。

3.2 経験ベースの検索機構

TAME経験ベースに対しては，各段階での多様な検索形態に対処するために，以下に示す一般的なデータベースの検索機構すべてが備わっている必要がある。

(1) キーワードに基づく検索

使用者から与えられたキーワードを含むモデル（データ）を検索する。

(2) ファセット指標（Faceted-index）に基づく検索

あらかじめ定められたファセットの各次元のデータ値を指定することでモデ

ルの集合を特定する。

(3) 意味ネットワークに基づく検索

関連づけられたモデルをその結合関係をたどることで検索する。

TAMEにおいて，キーワードに基づく検索が必要になる例として，経験の再利用が挙げられる。例えば，開発計算機がSunワークステーションであるプロジェクトをすべて列挙して欲しい，ある特定の語句を目的に含んだG/Q/Mモデルをすべて知りたいなどが挙げられる。

ファセット指標に基づく検索は，プロジェクトの定義段階で必要になる。TAMEにおいて，プロジェクトモデルはその各要素がファセット指標になったものとして定義されるので，過去のプロジェクトモデルの検索は，ファセット指標に基づくものになる。典型的な検索の例としては，まずキーワード検索でプロジェクトモデルの部分集合を得た後，その集合に対してファセット指標に基づく検索を行ない，必要なモデルの集合を特定する，という筋書きが考えられる。

意味ネットワークに基づく検索は，キーワードまたはファセットによる検索の結果として得られたモデルの集合の中から1つを特定するために必要である。また，特定したモデルから，経験則として関連付けられた別の種類のモデルをたどるさいにも，意味ネットワークに基づく検索が必要になる。

4．TAMEデータモデル

オブジェクト指向計算における計算モデル［SCH86，YON86］やデータモデル［BAN87，DEU90，FOR88，MAI86，ONT90，PUR87，WIL90］は言語やシステムによって多様な形態をなしている。これらは，共通点を数多く持ってはいるが，言語やシステム独自の機能を提供し，似て非なるものである。ここでは，これらの計算モデル，データモデルを包括的に調査し，TAME支援のためのデータモデルとして定義したものを紹介する。

4.1 オブジェクトとクラス

TAMEデータモデルで，オブジェクトは3.1で述べた種々のモデルの実体を表現するものである。オブジェクトには基底オブジェクト（Primitive Objects）と合成オブジェクト（Complex Object）の2種類がある。基底オブジェクトは整数，実数，文字，文字列など通常のプログラミング言語にみられる組込型（Built-in Types）または基本型（Basic Types）の変数に束縛された（bound）データオブジェクトに対応する。したがって，基底オブジェクトは値だけを持ち，その値の参照は基底オブジェクトに結合されたスロット（Smalltalk-80［GOL83］のインスタンス変数に対応するもの）

を介して行なう。

合成オブジェクトは使用者が定義するもので，SIMULA-67 [BIR80] のクラスから生成されるオブジェクトに対応する。合成オブジェクトはその内部状態を表すものとして複数の内部スロットを持つ。オブジェクト指向のもっとも重要な概念の１つであるデータ抽象化を実現するために，内部スロットはオブジェクト外部からは直接参照できないようになっている。内部スロットを参照または変更するためには，そのために定義されたメソッド（SIMULA-67 の手続きまたは関数，Smalltalk-80 のメソッドに対応する）をつねに介さなければならない。メソッドはメッセージをオブジェクトに送ることで起動される。メッセージに対応するメソッドは動的に決定される。これらの意味づけ（Semantics）は Smalltalk-80 のそれと同等である。

TAME データモデルにおけるクラスは SIMULA-67 のクラスに対応し，同一のオブジェクトをひとまとめにして取り扱う単位である。ここで同一とは，同じスロット群およびメソッド群を持ち，同じメッセージに反応することを指す。したがって，クラスはオブジェクトの型紙とも考えられ，後で述べるように，型付けの単位である。１つのクラスから複数のオブジェクトを生成できる。

4.2 クラススロットとプールスロット

TAME データモデルでは，各オブジェクトの内部状態を表す内部スロット以外に，複数のオブジェクトから参照または変更できるスロットであるクラススロットおよびプールスロットを導入した。これは，同種または異種の複数のモデルから参照または変更される情報を表現したいときに用いる。例えば，プロジェクトモデルにおいて，ある組織体に固有の人的資源の最大値などは，その組織体のプロジェクトではつねに同じ値をとる。このような情報を表現するのに，クラススロットまたはプールスロットを使用する。

クラススロットは同一クラスから生成されたオブジェクト群のどれからも参照および変更可能なスロットであり Smalltalk-80 のクラス変数に対応する。一方プールスロットは使用者が指定したクラス群から生成されたオブジェクト群から参照および変更可能なスロットであり Smalltalk-80 のプール変数に対応する。

4.3 is-a 関係によるクラス階層と継承機構

既存のモデル（クラス）のスロットやメソッドを特殊化したり，新しいメソッドやスロットを追加する等して，新しいモデルを作った場合，それらのクラス間に生じる一般－特殊関係を is-a 関係と呼ぶ。TAME データモデルでは is-a 関係によって一般－特殊関係を定義できる。これは３章で述べた同種のモデル同士を関係付けるために用いられる。is-a 関係によって結合された，同種のモデル群を同一ファイラム（phy-

lum）に属するモデル（クラス）と呼ぶ。また，is-a 関係で結合されたクラスで，一般的なほうをスーパークラス，特殊なほうをサブクラスと呼ぶ。

TAME データモデルでは ORION データモデル［BAN87］に見られるような一般的な多重 is-a 関係は導入していない。かわりに，ファセットに基づく多重 is-a 関係を導入した。これについては，5.2 節で詳述する。

is-a 関係はこのようなクラス間の論理的な関係を表現すると同時に，関係付けられたクラス間でコード共有の手段である継承を引き起こす。すなわち，is-a 関係で関係付けられたクラス群でサブクラスはスーパークラスから内部スロット，クラススロット，プールスロットならびにメソッドを継承する。メソッドおよび内部スロットはサブクラスで再定義が可能である。

一般的な多重 is-a 関係の上で継承を定義したものが多重継承であり，この場合は通常，同一名のスロットのかち合いのさいの規則を設けなければならない[BAN87]。さらに，論理的な is-a 関係を尊重した場合，物理的な多重継承と論理的な関係を分離する方策を構じる必要が生じる[LAL86, YAS91]。TAME データモデルでは，ファセットに基づく is-a 関係を定義したことにより，後で述べるように，これらの問題は解決された。

4.4 refers-to および has-a 関係とメソッドの戻り値およびスロットの型付け

TAME データモデルでは，すべてのスロット（内部スロット，クラススロット，プールスロット）およびメソッドの戻り値は型付けされる。この型システムは基本的なところでは Eiffel［MEY88］のそれと同じである。型付けの単位はクラスであり，あるクラスによって型付けされたスロットは，そのクラスまたはそのクラスのサブクラスのインスタンスである合成オブジェクトを値としてとることができる。

スロットの型付けは refers-to または has-a 関係を定義することで行なわれる。refers-to はオブジェクト間の参照関係である。すなわち，あるオブジェクトのあるスロットが別のオブジェクトを参照している場合，そのスロットは参照されたオブジェクトのクラスに型付けられる。refers-to 関係では，複数のオブジェクトのスロットから1つのオブジェクトを同時に参照できる。

他方，has-a はオブジェクト間の包含関係を示す。次節で述べるように，あるオブジェクトから has-a 関係で参照されているオブジェクトは部品であり，別のオブジェクトから同時に has-a または refers-to 関係で参照されない。参照しているオブジェクトを複合オブジェクト（Composite Object）と呼び，その内部スロットの少なくとも1つは has-a 関係で部品オブジェクトを参照する。そのさい，そのスロットは参照されたオブジェクトのクラスによって型付けられる。

TAME データモデルは，上述した通常の型付け機構以外に，制約された型付け機構

や動的型付けによるスロットの統一機構を提供する。これらについては，それぞれ，5.1および5.3で述べる。

4.5 複合オブジェクトとhas-a関係

TAMEデータモデルでは，オブジェクト間の全体ー部品関係を表す関係として，has-a関係を導入した。これは，3.1節で述べた工程モデルを記述するさい等に必要である。例えば，ある開発工程を全体をあらわす複合オブジェクトと定義し，その各段階を部品オブジェクトとして定義し，それらをhas-a関係で結合するという例が考えられる。複合オブジェクトの意味付けはORIONのそれと同じである。すなわち，他のオブジェクトとは排他的に占有する部品オブジェクトをもつものを複合オブジェクトと定義する。

5．TAMEデータモデルが提供する特徴的な機能

4章では，TAMEデータモデルを概説したが，本章ではTAMEデータモデルの特徴である，制約された型付け機構，ファセットに基づく多重is-a関係，スロットの動的統一機構について，詳しく述べる。

5.1 制約された型付け

これまでの型システムを持つオブジェクト指向言語［BIR80，MEY88，STO90，SCH86］やオブジェクト指向データベースのデータモデル［BAN87，DEU90，FOR88，MAI86，ONT90，PUR87，WIL90］においては，スロットの再定義に関してまったくそれを許さないか，メソッドやスロットの再定義に関して何ら制限を加えていないかのどちらかであった。

TAMEデータモデルではメソッドやスロットの再定義時の型付けに制約を課する形で再定義を許すことにした。筆者らが，過去のプロジェクトデータを経験ベースに格納しようと試みたとき，スロットの型付けに関してつぎのような制約が見られた：

(1) サブクラスのスロットの型はスーパークラスの同じ名前のスロットを型付けするクラスかそのサブクラスである。

(2) サブクラスのメソッドの戻り値の型はスーパークラスの同じ名前のメソッドの戻り値を型付けするクラスかそのサブクラスである。

(3) サブクラスのメソッドの引数の型はスーパークラスの同じ名前のメソッドの引数を型付けするクラスかそのサブクラスであり，引数の個数は同じである。

これは，サブクラスは本来スーパクラスに制約を加えることによって得られるものであることから考えると，きわめて自然な事実と考えられる。すなわち，スロットま

たはメッソドを型付けするクラス群と型付けされるクラス群のそれぞれの is-a 関係を適切に定義すれば，この型付けの関係は自然に守られるものだと考えられる。よって，筆者らはこの制約のついた型付けを TAME データモデルに採用することにした。

経験工場の立場からすると，制約された型付けは有用なものである。その理由は：

この型付け機構を用いれば，異なったファイラムのモデル群同士で is-a 階層に一致が見られるようになる，

からである。すなわち，より一般的な経験則は上位階層にあるクラス同士を結合することによって，より特殊な経験則は下位階層のクラス同士を結合することによって表現できることになる。これは，経験則の一般化に指針を与える基礎になる。すなわち，下位クラス同士の結合が頻繁に見られる場合は，それらを上位クラス同士の結合に置き換えることによって，経験を一般化して記憶する，等の法則を規定することが可能

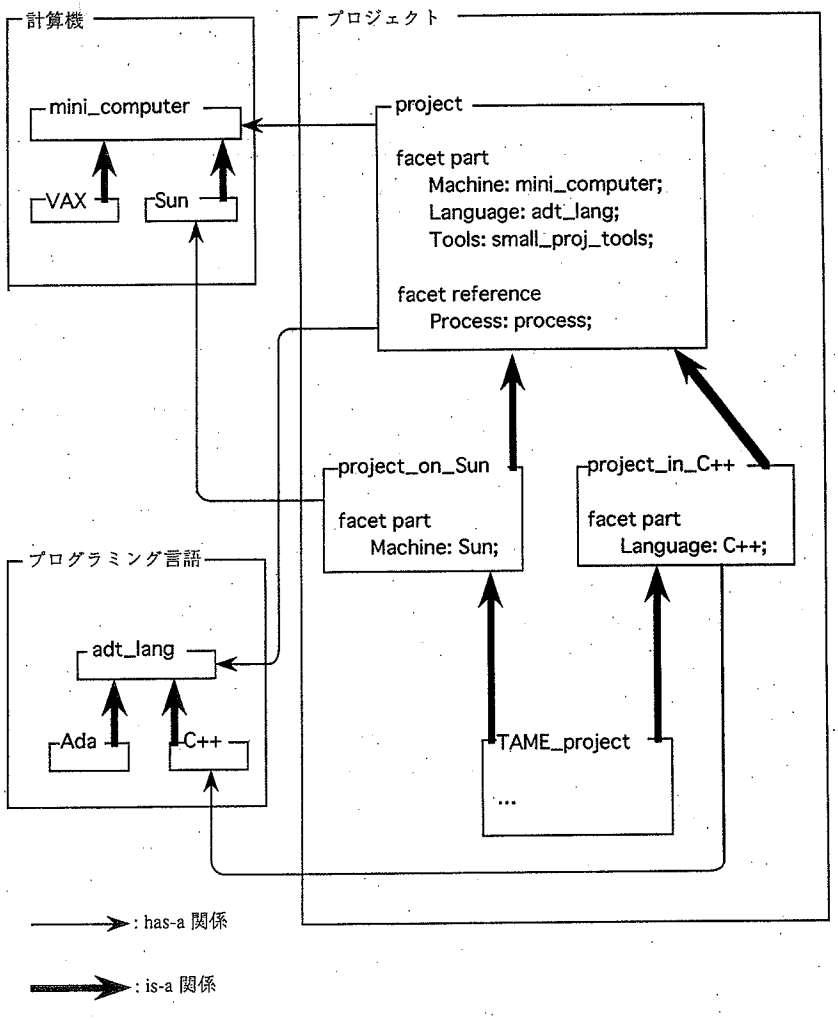

図 1　制約された型付機構と多重 is-a 関係

になる。

制約された型付けの例を図1に示す。図にあるように，プロジェクトファイラムの中のクラス project のスロット Machine は project_on_Sun で再定義されている。クラス project 中の Machine は mini_computer 型で宣言されているので，クラス project_on_sun の Machine は mini_computer，VAX，または Sun 型として宣言されなければならない。図1では，制約を守って，Sun 型として宣言されている。

5.2 ファセットに基づく多重 is-a 関係

筆者らは ORION に見られるような一般的な多重 is-a 関係を以下の理由から採用しなかった：

- is-a による論理的な一般－特殊関係と is-like-a [WEG87] による物理的なコード共有関係の混在により経験ベースの構造を混乱させないため。

  物理的なコード共有の関係は一般－特殊関係にあるオブジェクト間に継承という形で起こるものであって，物理的なコード共有が is-a 関係を規定するものではない。物理的なコード共有関係から導かれるものは is-like-a [WEG87] と呼ぶべき関係である。is-a 関係と is-like-a 関係を混同し，さらに無制限に多重 is-a 関係を取り入れることは，経験ベースのあるべき構造を破壊する恐れがある。これを避けるために，論理的な関係と物理的な関係を区別することが提案されている [LAL86, YAS91] が，それでもなお両者の関係を混同する危険は残されている。TAME データモデルでは下で述べるように，ファセットを表現するためだけに多重 is-a 関係を許すことにしたので，多重 is-like-a 関係の混同および多重 is-a 関係と多重 is-like-a 関係による経験ベースの構造の破壊の可能性は排除できた。だだし，単一 is-a 関係と単一 is-like-a 関係の混同によって，あるべき構造を破壊する可能性は残っている。しかし，これらは2つのオブジェクト間の関係であるので，その関係の本質（is-a か is-like-a か）を見極めるのは比較的容易である。

- 多重継承によるスロットやメソッドのかち合いの解決規則の考慮を避けるため。

  一般的な多重 is-a 関係を導入した場合，複数のスーパークラスで同じ名前のスロットやメソッドが宣言されているとき，そのかち合い解決(Conflict Resolution）の規則を適用しなければならない。多くのオブジェクト指向言語で，種々のかち合い解決の規則が提案されているが未だ一致した意見は存在しない。筆者らは，これは多重 is-a 関係を一般的すぎる形で採用したことに起因すると考え，制限をつけた形で多重 is-a 関係を採用した。

このように問題のある多重 is-a 関係を一切採用しないというのもデータモデル設計上の一選択ではあるが，

単一 is-a 関係だけを用いてファセットを表現するのは難しい

という問題がある。すなわち，単一 is-a だけを用いてファセットを表現するためには，ファセットの次元に順位をつける必要が生じる。この方法では，下位のクラスになればなるほど同じクラスを繰り返して定義する必要が生じる。次元の数がおおくなれば，組合せ爆発が起きるので，単一 is-a 関係だけでファセットを表現しようとするのは望ましくない。

以上のような議論を踏まえて，筆者らはファセットに基づく多重 is-a 関係を定義し，複合オブジェクトまたは合成オブジェクトのファイラム内でファセットを表現する場合に限り，多重 is-a 関係を使ってクラスの束（Lattice）構造を定義できるようにした。

定義と意味づけ

スーパークラスのあるスロットがファセットの指標として宣言された場合，サブクラスではそのスロットの型を特殊化できるだけで，他の属性を再定義してはならない。通常，ファセットの指標は複数個存在するので，それぞれのサブクラスではその内1個の再定義を行なえるだけである。このさい再定義は，ファセット指標を型付けするファイラムの is-a 関係を少なくとも部分的に保持する形で行なわれなければならない。このようにして定義されたクラスの束構造の葉の節点は，ファセットの次元を部分的に特殊化したクラスを複数個スーパークラスとして持ち，ファセットの次元が特殊化されたものになる。さらに 5.2 節で述べた制約された型付けの規則も守っている。

図1にファセットに基づく多重 is-a 関係の例を示す。図はプロジェクトモデルのファイラムを束構造として定義した場合を示す。ここでは，プロジェクトを特徴付けるファセットの次元として：

(1) 開発用計算機の種類，
(2) 開発に用いるプログラミング言語，
(3) プロジェクトで用いられる道具，
(4) 開発工程

の4つがあるものとする。図では，計算機と言語の次元を取り上げて，計算機としては VAX と Sun，言語としては，Ada と c＋＋が存在するものとする。最上位のクラス project では，上の4つの次元がファセット指標として宣言されている。そのサブクラスとして project_on_Sun と project_in_c＋＋が示されている。このような中間的なクラスは，各次元を規定するクラスの構造を反映して定義される。しかし，型付けする側の構造の全体を必ずしも模倣する必要はなく，通常その一部の構造を持つ。TAME_project はすべての次元の値が特殊化されたクラスであり，project_on_Sun, project_in_c＋＋，さらに project のサブクラスで他の次元を特殊化しているものすべてをスーパークラスとして持つ。

5.3 スロットの動的統一

スロットの動的統一とは，経験を表現するための refers-to 関係で関連付けられたクラス内の同じ名前を持つスロットの型を動的に統一することである。クラスａのあるスロット s1 がクラスｂによって型付けされている場合を考える。このときａとｂが同じ名前のスロット s2 を持っていれば，a. s2 の型と b. s2 の型はｂ型のオブジェクトを s1 に束縛したときに動的に統一され，a. s2 の型になる。

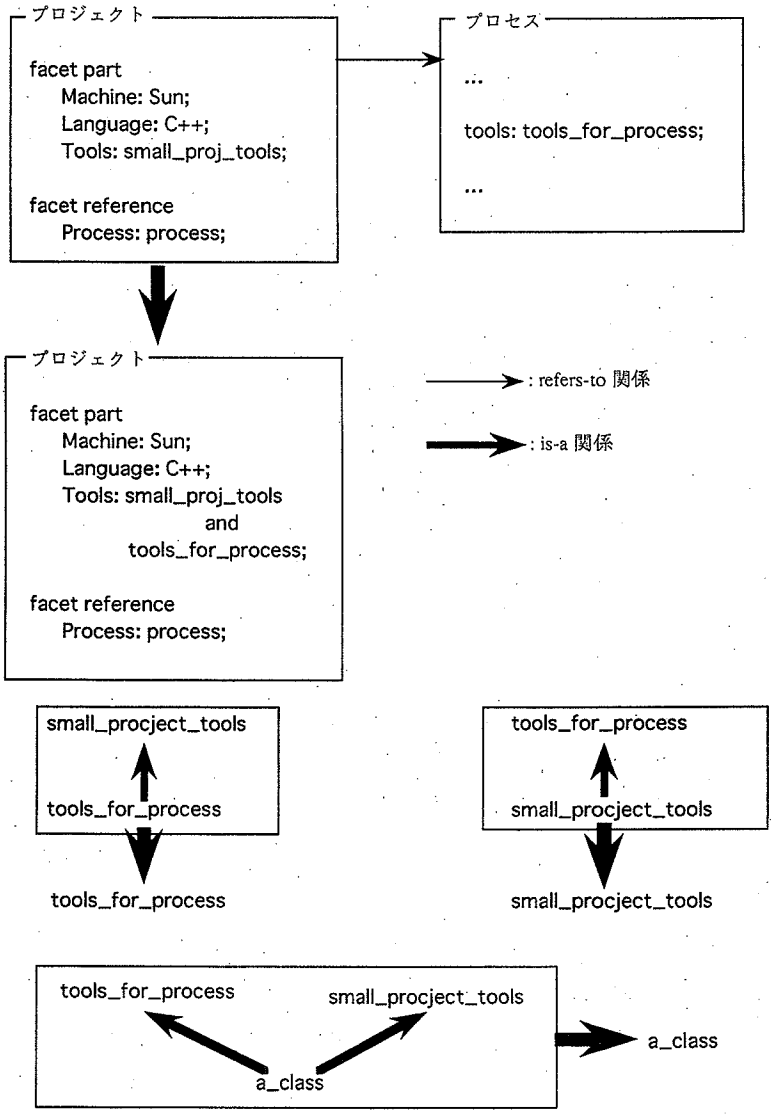

図２　オブジェクト属性の動的統一

型の統一の定義

いま，

class (s)：スロットsに束縛されたオブジェクトを型付けするクラス，

one_of_super (c1, c2)：クラスc1がis-a階層においてc2の上位に位置することを示す，

と定義したとき，型の統一は以下のように定義できる：

(1) one_of_super (class (a. s), class (b. s))ならば結果の型はclass (b. s)。

(2) one_of_super (class (b. s), class (a. s))ならば結果の型はclass (a. s)。

(3) あるクラスcについて，one_of_super (class (a. s), c)かつone_of_super (class(b. s), c)ならば，結果の型はこのようなcのうちis-a階層の最上位に位置するもの。

(4) それ以外の場合はエラー。

動的な型の統一の必要性は，具体的な例を考えれば理解できる。いま図2に示すように，クラスTAME_projectがクラスprototypingをその開発工程として参照しているとする。さらに，両クラスともスロットToolsを持つものとする。これらのクラス間のrefers-to関係が示すものは，“TAME_projectがprototypingを開発工程として成功した”という経験則である。このときTAME_projectで使用する道具のもっとも一般的な組がclass (TAME_project. Tools)であり，prototypingを遂行するために必要な道具のもっとも一般的な組がclass (prototyping. Tools)である。この型付けの意味するところは，これらの組以上に一般的な道具の組ではプロジェクトを遂行できないということである。動的統一を行なうと，これらのクラスで定義される特殊な道具の組のなかでもっとも一般的な組の型を規定することになるので，この経験則が成り立つならば統一された型の道具を使ってプロジェクトが遂行されたことになる。

6．経験ベースの実現状況

4，5章で述べたデータモデルに基づいて，筆者らは経験ベース管理システムを試作した。そのTAMEシステム内での位置づけは図3に示すとおりである。データモデルの実現は実験的フレームシステムXRL [CHA87]を拡張することによって行なった。これをXRL＋＋と呼ぶ。XRL＋＋の実現言語としてCommon LISPを用いSun UNIX上で試作を行なった。

XRL（＋＋）はオブジェクトの永続性（persistence）については考慮していないので，筆者らはObSERVer [SKA86]を用いて，オブジェクトサーバ（オブジェクトを情報の最小単位として補助記憶に保持する）を実現している。さらに，収集したデー

タを保持しておくために関係データベースORACLEを用い，ORACLEとXRL＋＋のインターフェースシステムを作成した。

工程モデルなどXRL＋＋以外の言語で記述する必要のあるモデルに関しては，その言語で書かれたテキストとして，XRL＋＋のスロットに保持する。これらテキストの処理のために，メソッドを定義し，そのメソッドからその言語の処理系を起動するという方式で他言語への対応を実現している。すなわちデータモデルの設計目標である開放性を実現するために，XRL＋＋からUNIXのコマンドを実行しその結果をXRL＋＋に取り込むメカニズムをCommon LISPのOSインターフェースを用いて実現した。

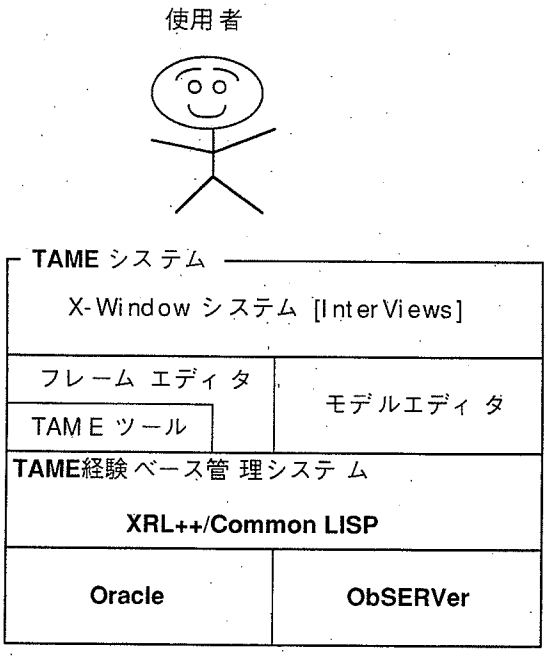

図3　TAMEシステムと経験ベース

7. おわりに

筆者らは品質改良パラダイムとG/Q/Mパラダイムに基づくTAMEを提案し，その過程での作業を可能なかぎり計算機で支援するTAMEシステムの実現を目指している。その第一歩として，TAME経験ベースの設計および試作を手掛けた。本論文では，経験ベースのデータモデルについて述べた。このデータモデルは従来のオブジェクト指向データモデルを拡張したもので，従来のデータモデルにはない，

- 制約された型付け，
- ファセットに基づく多重is-a関係，および

・スロットの動的統一

を規定している。これらの新しい機能は従来からあるオブジェクト指向データベースのデータモデルに制約を加えることによって得られたものである。

今後の研究課題として以下のものが残されている：

(1) スキーマ進化（Schema Evolution）に関する諸問題の解決

現在，XRL＋＋による経験ベースの設計・実現においては，スキーマ進化を考慮していない。本論文で述べたTAMEデータモデルの特徴的な機能を取り入れたことにより，スキーマ進化の取り扱いはORIONほど簡単ではなくなる。今後この点について，議論を重ね解決していく必要がある。

(2) 拡張可能なデータベース管理システムの設計と実現

試作した経験ベースを使って，幾つかの組のプロジェクトデータを記述してみた結果，オブジェクト間の関係はis-a, has-a, refers-toだけでは不十分であることがわかった。むしろ，必要なオブジェクト間の関係はプロジェクト毎に異なると仮定するほうが自然であると考えられる。この考えに基づき，オブジェクト間の関係をカスタマイズ可能で拡張性のあるオブジェクト指向データベース管理システムLeO [MAE92]を設計・実現することとした。LeOではその記述言語Liyaによってオブジェクト間の関係を定義できるだけでなく，データベース管理システムとして必要な，トランザクション処理やスキーマ進化についても変更可能である。今後LeO上にデータモデルを実現し，ソフトウェア開発プロジェクトデータを保持することを試みることでその改版を重ねていきたい。

参考文献

[AND87] Andrews, T. and Harris, C.: Combining Language and Database Advances in an Object-Oriented Development Environment, Proc. ACM Conf. on OOPSLA, pp. 430-440 (1987).

[BAN87] Banerjee, J. et al.: Data Model Isses for Object-Oriented Applications, ACM Trans. Office Info. Sys., Vol. 5, No. 1, pp. 3-26 (1987).

[BAS88] Basili, V. R. and Rombach, H. D.: The TAME Project: Towards Improvement-Oriented Software Environments, IEEE Trans. Softw. Eng., Vol. SE-14, No. 6, pp. 758-773 (1988).

[BAS89] Basili, V. R.: Software Development: A Paradigm for the Future, Proc. 13th Annual Int. Comp. Softw. & Applications Conf., pp471-485 (1989).

[BIR80] Birtwistle, G. M. et al.: SIMULA Begin, Studentlitteratur, Lund: Sweden (1980).

[CHA87] Charniak, E. et al.: Artificial Inteligence Programming, 2nd ed., Lawrence Erlbaum Associates, NJ: Hillsdale (1987).

[DEU90] Deux, O. et al.: The Story of O2, IEEE Trans. Knowledge Data Eng., Vol. 2, pp.

91-108, (1990).

[FOR88] Ford, S. et al : Zeitgeist : Database Support for Object-Oriented Programming, Proc. 2nd Int. Workshop on Object-Oriented Database Syst., pp. 23-42 (1988).

[GOL83] Goldberg, A. and Robson, D. : SMalltalk-80 : the Language and its Implementation, p. 714, Addison-Wesley Pub. Co., MA (1983).

[LAL86] LaLonde, W. R., Thomas, D. A. and Pugh, J. R. : An Exemplar Based Smalltalk, Proc. ACM Conf. on OOPSLA, pp. 322-330 (1986).

[MAE92]前田和昭, 野呂昌満, カスタマイズ可能なソフトウェア開発支援環境用データベース管理システムの設計：自己反映機能を利用して, ソフトウェア技術者協会シンポジウム論文集, (1992).

[MAI86] Maier D. et al. : Development of an Object-Oriented DBMS, Proc. ACM Conf. on OOPSLA, pp. 472-482 (1986).

[MEY88] Meyer, B. : Object-Oriented Software Construction, p. 534, Prentice Hall, (1988).

[ONT90] Ontologic Incorporated, Ontos System Documentation, Billericam MA : Ontlogic, Inc. (1990).

[SCH86] Schaffert, C. et al. : An Introduction to Trellis/OWL, Proc. ACM Conf. on OOPSLA, pp. 9-16 (1986).

[SKA86] Skarra, A. Zdonik, S. and Reiss, S. : An Object Server for an Object-Oriented Database System, Workshop on Object-Oriented Databases, Pacific Grove, CA (1986).

[STO90] Stonebraker, M., Rowe, L. and Hirohama, M. : The Implementation of POSTGRESS, IEEE Trans. Knowledge Data Eng., Vol. 2, pp. 125-141, (1990).

[STR91] Stroustrup, B. The C++ Programming Language 2nd Ed., p. 669, Addison-Wesley Pub. Co., MA (1991).

[WEG87] Wegner, P. The Object-Oriented Classification Paradigm in Research Directions in Object-Oriented Programming, Shriver, B. and Wegner, P. (ed), MIT Press, pp. 479-560, (1987).

[WIL90] Wilkinson, K., Lyngboek, P. and Hasan, W. : The Iris Architecture and Implementation, IEEE Trans. Knowledge Data Eng., Vol. 2, pp. 63-75, (1990).

[YAS91] Yaseen, R., Su, S. Y. W. and Lam, H. : An Extensible Kernel Object Management System, Proc. ACM Conf. on OOPSLA, pp. 247-263 (1991).

[YON86] 米澤明憲ほか, オブジェクト指向に基づく並列情報処理モデル ABCM/1 とその記述言語 ABCL/1, コンピュータソフトウェア, Vol. 3, No. 3, pp. 9-23 (1986).