存在従属性によるオブジェクト識別子の設計
Designing Object Identity by using Existence Dependency

井田明男† 金田重郎†
Akio IDA Shigeo KANEDA
†同志社大学・理工学部
†Faculty of Science and Engineering, Doshisha University.

要旨
オブジェクト指向では伝統的にオブジェクト識別に単一属性であるオブジェクト ID（OID）が用いられてきた．しかしながら，OID だけでドメインモデルを検討することはデータの整合性要件に対して脆弱なモデルを生む．そこで，本稿では，存在従属性に着目した識別子の付与方式とクラス図をベースとした表記法を提案する．存在従属性の概念は理解しやすく，システム化のドメイン中のいたるところで見出されるため，提案手法により，第 5 正規形と同じ正規化レベルを持ったドメインモデルが無理なく構築できる．

1. はじめに

今日，ビジネスアプリケーション開発では経営環境の急速な変化に対応するために空前のアジリティが求められている．同時に，多くの組織で開発しているシステムは，昨今のビッグデータブームからも明らかなように，よりデータ指向的になってきている[1]．そして，ビジネスアプリケーションの世界では，ドメインモデルの構築にはオブジェクト指向アプローチ（OOA）と UML 表記を採用し，その実装には Java のようなオブジェクト指向プログラミング言語（OOPL）と Oracle のような関係データベース管理システム（RDBMS）を用いた開発が一つの典型的なスタイルとなっている[2].

しかしながら，OOA の発想だけでは，論理要件に対して頑健なドメインモデルを構築することは難しく，また UML のクラス図表記もそのままでは論理要件を表記し難いという問題がある．なぜならば，OOA ではどのようなクラスのインスタンスであっても，その識別子として属性とは別に単一属性のオブジェクト ID（OID）を仮定するためであり，クラス図にはわかり切った OID を含めて主キーを明示的に表記しない慣習があるためである．このような識別子の取扱い方では，クラス間の結び付きの意味が曖昧になり，業務ドメインにおいて重要なデータの整合性要件を満足にモデルに表現することはできない．結局，モデルに表現されない要件は正しく実装されないか，あるいは忘れ去られる運命にある．そのような理由から，かつてのデータ中心アプローチ（DOA）の手法を見直す動きもある[1]が，直感的な分りやすさと収束の速いトップダウンの実体主導型が身上の OOA はアジャイル開発に適しており，ここに急遽 DOA のようなデータ項目主導型の正規化理論を持ち込むことは困難であると考えられる．

そこで，本稿では，OOA とクラス図をベースに存在従属性に着目したドメインモデルの構築手法と表記法を提案する．この手法は，クラスの識別子とクラス間の関連をインスタンスにとって生まれながらの存在従属性と後天的な参照を峻別して見直すものである．提案手法を適用すれば正規化理論を表に出さずに同等の論理要件に頑健なドメインモデルを迅速かつ正確に構築できる．ただし，存在従属性の概念を紹介した論文[3]やこの概念をモデリングに導入する提案[4]は過去にも存在するため，本稿ではこの概念をオブジェクトの属性や識別子の決定に用いることを，その表記と併せて提案する．

以下，第 2 章では，本提案に関係する概念の定義と関連研究の概要を述べる．第 3 章では，本提案の手法を説明する．第 4 章では，本提案手法を例題に適用した結果を議論する．第 5 章はまとめである．

2. 概念の定義と関連研究

2.1. オブジェクト指向と正規化理論

表 1 は DOA と OOA を大雑把に比較対照したものである．DOA には正規化と呼ばれる集合論に裏打ちされた設計理論が存在することを示す．しかしながら，OOA にはそれに相当するような方法は未だ存在しない．それでは，OOA にも正規化理論を導入すればよいと考えられるが，前述の通り OOA の識別子の考え方は正規化理論に馴染まない．OOA における識別子の考え方は，先にオブジェクトを識別し，すべてのオブジェクトには生まれながらにして，OID と呼ばれる単一属性の識別子が自動的に付与されるとされる（以下，OID 方式と表現する）．一方，DOA においては，アウトプットに求められる属性項目を，正規化理論を用いて，結合従属性を失わないように結合ある

いは分解する．したがって, DOA における識別子の考え方には, 複合主キーが自然かつ頻繁に登場する．たとえば, 第 1 正規形から第 2 正規形へ変換する際には，識別子に部分関数従属しているデータ項目を分離するが，そもそも部分関数従属という概念は，複合主キーが前提である．OOA の識別子の考え方と，DOA の識別子の考え方は根本的に異なる．

表 1　DOA と OOA の大局的な対比

項目	DOA	OOA
着眼点	属性（項目）主導型	実体主導型
アプローチ	ボトムアップ中心	トップダウン中心
モデリングプロセス	正規化	直感方式
識別子	複合主キー前提	オブジェクトIDと呼ばれる単独識別子を常に仮定

ここで，ユーザの要求を充足するアプリケーションが提供できるのであれば，いずれの識別子の考え方に基づいていてもよいが，オブジェクトの永続化の実現手段として，90%以上 RDBMS が用いられている現状[2]においては，事情が異なる．OOA の識別子の発想では，整合性要件を充足するアプリケーションの構築が難しい．なぜならば，OID 方式には，複合識別子は登場しない．クラスがどのようなタイプのものであれ，すべてのインスタンスは単一属性の OID で識別されることになっている．このような識別子の付与を行った場合，たとえ優れた RDBMS を永続化層で使用していたとしても, 整合性制約をアプリケーション側で実装しなければならないモデルになる. 困ったことに, 前者のアプローチで作成されたモデルと後者のアプローチで作成されたモデルは，一見非常によく似ている．そして，クラス図には伝統的に主キーが表記されないことが事態をより深刻にしている．データの整合性要件（業務の論理要件）の観点からは，OOA のモデルは脆弱になりがちである．このことは，ドメインモデルの作成とデータモデルの再作成という重複する努力を排除できない原因の一つとなっている．

2.2. 存在従属性

存在従属性の概念は P．チェンが用いている[5]．あるオブジェクトが，別のオブジェクトの先立つ存在を前提として存在し得るとき，前者のオブジェクトは後者のオブジェクトに存在従属するという．そして前者のオブジェクトを**従属クラス**（dependent class）のオブジェクト，後者のオブジェクトを**独立クラス**（master class）のオブジェクトと呼ぶ[4]．この存在従属という概念を用いることによって，たとえば，ある区間は，異なる２つの地点に存在従属する，月々のローンの返済は，過去の購買という事象に存在従属する，などと表現できる．スリーアミーゴスの OOA の文献ではわずかに[6]だけが存在従属性について言及している．

2.3. 存在従属性と属性

存在従属性は，オブジェクト間だけの関係ではない．オブジェクトの属性の間にも存在従属性が認められる．すなわち，属性とは，あるオブジェクトが生まれるとともにそのオブジェクトの値として生まれ，そのオブジェクトがこの世に存在する限り，そのオブジェクトに付随し，そのオブジェクトがこの世から消えるとともに当該属性も消えるデータ項目のことである．したがって，属性群はオブジェクトに存在従属するといえる．言い換えれば,「オブジェクトに存在従属するデータ項目こそが属性」という観点でそれらを定義すると，属性はオブジェクトに完全関数従属するとともに，候補キーではない属性から候補キーを構成する属性への関数従属性をも排除することができる．クラスが持つべき性質で必要なのは,「すべての事実は，キーだけに関係する事実である[7]」を満たすため, これだけでも少なくともボイスコッド正規形が達成される．次章では OOA の単純さ，直感的な分りやすさを犠牲にすることなしに，論理要件に対して頑健なドメインモデルを構築する手法とその表記法を提案する．

3. 提案手法

提案の核心は極めてシンプルである．具体的には存在従属性の概念を導入したドメインモデリングのガイドラインを次のように定める．

イ） あるデータ項目をあるクラスの属性とするには，その値がそのクラスのインスタンスに存在従属する場合に限る.

ロ） そのインスタンスが，独立して存在可能なクラスの ID には「必ず」単一属性の ID を与える.

ハ） そのインスタンスが，独立して存在できない（存在従属な）クラスの ID には,「決して」単一属性の ID は与えず,「必ず」その存在根拠となるクラスの ID を含む ID を与える．結果，存在従属なクラスの識別子は複合主キーとなる

ニ） そのインスタンスが，時点が帰属するイベント[8]の場合は，上記のルールに加えて，そのクラスの ID に「必ず」時点をあらわす時刻または版（バージョン）のシリアル値を含める.

ホ） 存在従属でないクラス間の関係は，汎化関係と単なる参照関係のみとする．

i． 汎化関係の場合：サブクラスの ID は「必ず」スーパークラスと同じ ID を共有する．たとえば，「プレミアム会員」のスーパークラスが「会員」クラスで，その ID が「会員 ID」であった場合は，「プレミアム会員」クラスの ID も「会員 ID」となる．

ii． 単なる参照関係の場合：参照元のインスタンスと参照先のインスタンスのそれぞれのライフサイクルは独立している．このような場合は，表記では参照元のクラスから参照先のクラスへの依存関係を定義し，RDB では参照元は参照先の ID を外部キーとして保持するものとする．

以下では，このガイドラインを用いた手法について説明する．例題として，専門学校などで，１週間の時間割の策定を支援するアプリケーションを考える．このアプリケーションには，さまざまな制約がある．制約とは，「実習科目に座学教室が割り当てられぬこと」など，必ず真にならなければならない条件である[7]．ドメインの専門家（たとえば学校の時間割策定担当者）は，このような成立して当然の条件群は，通常わざわざ口にしないと考えられるが，それでも制約は論理要件として厳然と存在するので，ドメインモデルではそれらを表現しなければならない．ある学級の月曜日の１時限目の授業を考える．学級と曜日と時限が決まれば，授業が決まる．おのおのの授業には，講師，教室，そして教科が必要である．しかし，教科によって担当できる講師や教室の要件（座学教室か実習教室か）に制約がある．また，講師も非常勤の場合は，出講（登壇）可能な曜日と時間帯に制約があるのが普通である．さらに，教科もその学級で１週間で実施すべき教科が過不足なく網羅されなければならない．こうなってくると OOA の発想だけでは，論理要件に頑健なドメインモデルを構築することはもはや難しいはずである．けれども，OOA の発想に存在従属の概念を追加するだけで，この問題は劇的に考えやすくなる．以下では，その手順とモデルの表記について説明する．

3.1. 手順とモデルの表記

3.1.1 Step1：当該ドメインにおける独立クラスを網羅し，それぞれに識別子を付与する

まずは独立クラスを識別する．独立クラスとは，当該ドメインにおいてそのインスタンスが単独で存在できるクラスである．この問題の場合，学級，教科，講師，教室，曜日，時限が独立クラスである．ここで或いは，学級は，その学級が置かれる学校に存在従属するような従属クラスではないかと考える方もおられるかも知れないが，ドメインがその学校一校に限られるのであれば学校そのものをモデル化する必要はない．

独立クラスには，前述のガイドライン（ロ）にしたがって，その識別子として単一属性の主キーを付与しなければならない．例題では，学級 ID，教科 ID，講師 ID，教室 ID，曜日 ID，時限 ID をそれぞれ対応する独立クラスの識別子として付与する．この付与法は OID 方式と同様である．

図 1 は Step1 実施後のドメインモデルである．図 1 では，モデルを簡潔に保つために主キー属性を示す目的でステレオタイプは用いず，属性名の後に「ID」を追加してそれを示すことにした．

3.1.2 Step2：独立クラスのインスタンスに存在従属するデータ項目（すなわち属性）を記述する

つぎに，独立クラスのインスタンスに存在従属するデータ項目（すなわち属性）を記述する．データ項目はすべて網羅する必要はなく，業務要求に照らして主要なものを 2～3 個記述するだけでよい（クラスが識別された後で充

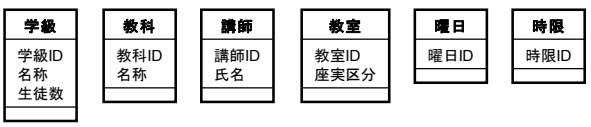

図 1　例題ドメインの独立クラス群

図 2 属性が追加された例題ドメインの独立クラス群

実させればよい）．こうすることによって，そのクラスの存在意義を具体的に把握しやすくなる．ここで大切なのは，上述のガイドライン（イ）にしたがって，データ項目は当該インスタンスに存在従属するものだけに限定することである．データ項目のインスタンスに対する存在従属性を保つことにより，そのデータ項目は文字通り「属性」となるため，正規化理論でいうボイスコッドの正規形が自然と満たされることになる．

例題では，たとえば，教室クラスに，座実区分が置かれている．その教室が座学教室なのか，実習教室なのかは教室に存在従属するデータ項目と考えられるからである．図 2 は Step2 実施後のドメインモデルである．

3.1.3 Step3：当該ドメインの独立クラスに存在従属するクラスを網羅する

今度は，従属クラスについて考えていく．従属クラスとは，当該ドメインにおいて，そのクラスのインスタンス

が単独で存在することができないクラスであり，必ず別のインスタンスの先立つ存在（生成済み）が前提である．この問題の場合，講師の「担当できる教科」は，講師と教科に存在従属する，「講師の出講できる時間帯」は，講師と曜日と時限に存在従属するなどと考えながら従属クラスを発見していくことができる．

図 3 は Step3 実施後の例題のドメインモデルである．図 3 では，前述のガイドライン（ハ）にしたがって，各クラスの識別子としてその存在根拠（前提）となるクラスの ID を含む ID を与えている．その結果，従属クラスの識別子は複合キーとなっている．たとえば，担当可能クラスの識別子は，教科の ID と講師の ID の連結した複合主キーである．こうすることによって，担当可能クラスのインスタンスは必然的に，実存する講師と実存する教科の組み合わせに限定されるため，正規化理論でいう第 5 正規形が自ずと満たされることになる．

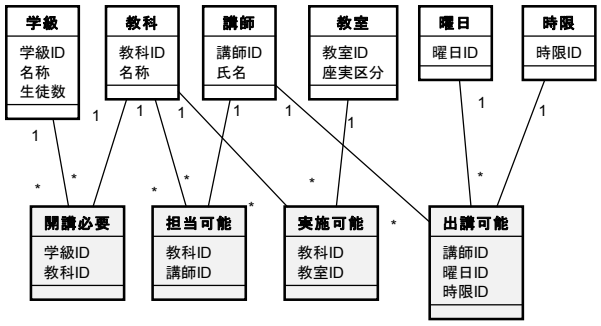

図 3　Step3 実施後の例題のドメインモデル

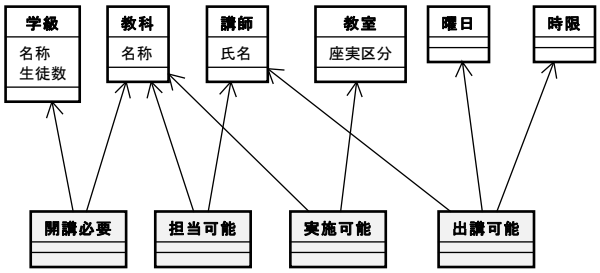

図 4　図 3 の提案記法による表記

ここで，表記法についても検討したい．図 3 では，従属クラスを淡い灰色で表記し，おのおののマスタークラスとの間に関連を定義しているが，その関連が存在従属であることをモデルの読み手に強く印象づけたい．なぜならば，関連の意味が存在従属であることが正しく伝われば，従属クラスの複合識別子をわざわざ表記する必要がなくなり，モデルが簡潔になるからである．同時に，従属クラスからみた独立クラスの多重度は必ず 1，従属側は多になっているため，多重度もわざわざ表記する必要はなくなる．そこで，図 3 のモデルは図 4 のように表記することも併せて提案する．図 4 では，上述のガイドライン（ロ）が徹底されるのであれば，独立クラスの識別子でさえも省略可能であることを示唆している．

さて，図 4 では関連の種類が存在従属性であることを表現するために誘導可能性を使用している．ドメインモデルは，ユースケース実現のモデリングを実施する以前のモデルであるため，誘導可能性がモデルに登場することはないため，このモデル要素を流用することにした．当初はコンポジションを用いることも考えた．コンポジションは，たとえば，注文クラスと注文明細クラスの関係を表現する場合によく用いられる（図 5）．

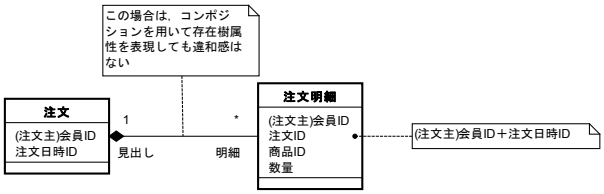

図 5　コンポジションによる存在従属性の表記（適切）

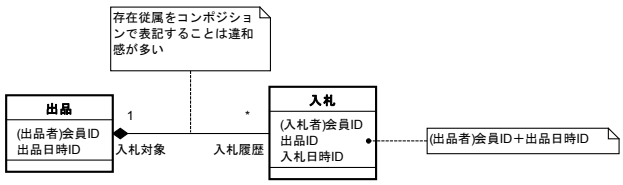

図 6　コンポジションによる存在従属性の表記（不適切）

確かに，注文明細は注文に存在従属するし，注文が全体側で，注文明細は他の注文と共有できない部分側のインスタンスである．このような見出し－明細パターンに見られる存在従属性を表現する場合はコンポジションが相応しい．しかしながら，別の例，たとえば，オークションサイトの入札は，あきらかに出品に存在従属する（存在しない出品に対する入札は不可能であるため）が，この関係を図 6 のように，コンポジションを用いて表記した場合は，入札は出品の部分であるとは考えにくいため違和感が残る．コンポジションは，全体と部分の関係（しかも全体側にとって部分側を共有できない）という意味[9]の方が強調されてしまうように感じられる．そこで，コンポジションを存在従属の表記に使用することはいつも適切であるとは限らない．

ここでもう一つ注意事項がある．それは，あるクラスから見た別のクラスの多重度が厳密に 1 であるからといっ

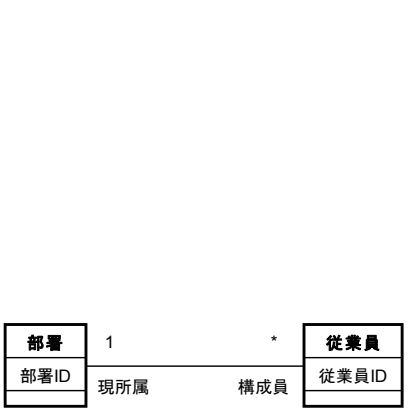

図 7　存在従属ではない関連の例

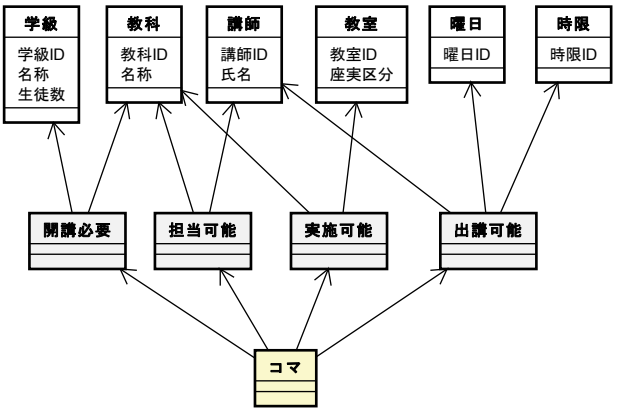

図 8　例題ドメインの Step4 実施後のモデル

て, 前者のクラスは後者のクラスに存在従属しているとは限らないことである. 図 7 はそのような場合の例である. このモデルは, 従業員が部署に存在従属することを表しているのではなく,「従業員たるものは常時どこか 1 つの部署に所属しなければならない」という業務ルールを表わしている.

3.1.4 Step4：当該ドメインにおける従属クラスに存在従属するクラスを識別する

さらに，当該ドメインにおける従属クラスに存在従属するクラスを識別する．例題では，時間割のそれぞれのコマがそれに該当する．図 8 は「コマ」クラスをドメインモデルに追加したものである.

図 8 のモデルでは，時間割の各コマが満たすべき論理要件が正確に反映されている．すなわち，コマのすべてのインスタンスは，ある学級において開講が必要な教科について，その教科が担当可能でかつその時間帯に出講可能な講師と，その教科を実施可能な教室の先立つ存在を前提として存在できることを示している．これは，時間割策定支援アプリケーションの論理要件そのものである上，このモデルは 3.1.3 節で確認した通り，第 5 正規形が満たされている．表 2 は提案手法が正規化にどのように寄与しているかについてまとめたものである．正規化レベルごとの概要については渡辺[10]を参照した．渡辺は，第 5 正規形は努力目標ではなく，このレベルでないデータモデルは使い物にならないと述べている.

4. データモデルによる検証

前章では，存在従属性の概念を用いたドメインモデルの作成について，やや極端な例題を用いてそのガイドラインと手順，およびモデルの表記について説明した．独立クラスと従属クラスを意識することで，ドメインの理解が促進される上，その論理要件を満たす理想的な識別子について考慮も無意識に実施してしまえることが明らかとなった．この章では，この例題のデータモデルを用いて提案手法を検証する．図 9 は，図 8 のドメインモデルの頑健性を検証するために作成したデータモデルの例である．ドメインモデルから直接 MySQL を使って実装し，試しにいくつかの具体値を用いてデータの追加，更新，削除を行ってみた.

4.1. 試行結果

試行の結果，このモデルは，次の当り前の業務要件を愚直にクリアし，時間割策定支援のアプリケーション側の整合性検査ためのロジックを大幅に減らした.

- 存在しないリソースの組み合わせは決して登録できない
- 存在しない組み合わせからなる組み合わせも決して登録できない
- あるリソースが使われる組み合わせが存在している限り，そのリソースは削除できない
- ある組み合わせが使われている組み合わせが存在している限り，その組み合わせは削除できない

5. まとめ

本稿では，ドメインモデル作成のために存在従属性に着目したモデリング手法と表記法を提案した．ドメインモデルがドメインの専門家と開発の技術者が共通かつ厳密に意思疎通可能なユビキタス言語として機能するためには，そのガイドラインにおいても表記においてもノイズを排除した究極のシンプルさと，論理要件に対する頑健さを兼

表 2　提案手法の正規化レベルへの寄与

正規化レベル	概要	提案手法における対応
第1	非キー属性のうち繰り返しデータ項目などの集合要素を分離	対応なし．しかし，繰り返す属性は，発見に労を要しないため，発見次第直ちに別の実体として分離すれば問題なし
第2	非キー属性のうち，主キーの一部だけに関数従属（部分関数従属）するものを分離	実体に存在従属するデータ項目のみを属性とするため，当初から分離される
第3	非キー属性のうち，主キーから推移的に関数従属するものを分離	実体に存在従属するデータ項目のみを属性とするため，当初から分離される
ボイスコッド	非キー属性のうち，候補キーに対して部分あるいは推移関数従属するものを分離	実体に存在従属するデータ項目のみを属性とするため，当初から分離される
第4	関係から多値従属性のある関係を取り出して分離	このような関係は従属クラスとして識別されるため，当初から分離される
第5	関係のうち，結合従属性がある関係を 取り出して分離	このような関係は従属クラスとして識別されるため，当初から分離される

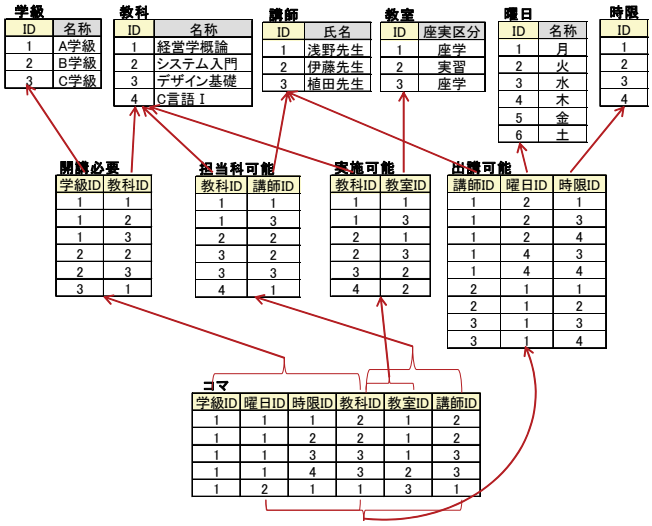

図 9　検証のために用いた例題のデータモデル

ね備えることが望まれる．存在従属性の概念とそれに関わる表記もすでに存在しているが，存在従属性を OOA において正規化理論に相当するものと位置づけたこと，そして表記を世界標準の UML のクラス図に結びつけたところに本提案のオリジナリティがある．

存在従属性の概念は理解しやすくドメイン中で見極めやすいため，提案手法により，第 5 正規形と同じ正規化レベルを持ったドメインモデルが容易に構築できる．これは RDBMS を単なる永続化層としてではなく，データの整合性を保つための砦として使用できることを意味する．

筆者らは，予てから日本語問題記述からドメインモデルを工学的に導く研究を進めている[11][12]が，これらの研究に今回の提案を加えることで，さらにモデル化の精度を高められると確信している．提案手法がオブジェクト指向手法において正規化と同等の役割を果たし，ドメインエキスパートとシステムエンジニアが共同で参加するモデリング作業の一助となれば幸いである．

参考文献

[1]　一般社団法人 ICT 経営パートナーズ協会：超高速開発が企業システムに革命を起こす，日経 BP 社 (2014)

[2]　南波幸雄：企業情報システムアーキテクチャ，翔泳社(2009)

[3]　M．Snoeck，G．Dedene：Existence dependency: The key to semantic integrity between structural and behavioral aspects of object types，　IEEE Transactions of Software Engineering，24(4)，pp 233-251 (1998).

[4]　R．Haesen，M．Snoeck，W．Lemahieu，S．Poelmans：Existence dependency-based domain modeling for improving stateless process enactment，Hogeschool-Universiteit Brussel, Belgium (2009)

[5]　P．Chen：The Entity Relationship Approach to logical Database Design，QED information science, Wellesley (1977)

[6]　James Rumbaugh：オブジェクト指向方法論 OMT―モデル化と設計，トッパン (1992)

[7]　C．J．Date：Database In Depth，Oreilly & Associates Inc (2005)

[8]　佐野正美：論理データベース論考，ソフト・リサーチ・センター (2000)

[9]　OMG：UML Superstructure Specification, v2.4.1，http://www.omg.org/spec/UML/2.4.1/

[10] 渡辺幸三：データモデリング入門，日本実業出版社 (2001)

[11] 金田重郎，井田明男，酒井孝真：英語 7 文型と関数従属性に基づくクラス図の理解，電子情報通信学会，知能ソフトウェア研究会 (2014 年 3 月)

[12] 井田明男，金田重郎：オブジェクトへの責務配分のための多次元尺度構成法の適用～ソフトシステムズ方法論の成果物からの接近～，電子情報通信学会，知能ソフトウェア研究会 (2014 年 3 月)