Vol.2016-DBS-163 No.19
Vol.2016-IFAT-123 No.19
2016/9/15

オブジェクトのライフタイムに基づくクラス図の理解

金田 重郎[1,a)]　井田 明男[1]

概要：UML は，クラス図の「関連 (association)」を「意味論的な関連性を示す」と定義している．しかし，この定義では，エンジニアの具体的設計指針とするのは難しい．この課題を解決するため，本論文では，「クラス図の『関連』は関係データベースの関数従属と等価であるべき」，との視点に立脚する．そして，この観点からクラス図の構成を理論的に分析する．関連が関数従属と等価となるには，関連の片側の多重度は「1」となる必要がある．その場合，多重度 0 は，1) オブジェクトの生成タイミング，又は，2) オブジェクトのライフタイムを，関数の入力側オブジェクトの属性として持たせることにより表現できる．更に，関数従属の関連は，ライフタイムの短いオブジェクトから，長いオブジェクトにのみ張り得る．以上の理論的分析から，関数従属の「関連」を持つクラス図の構成が規定される．本提案によるクラス図の構成は，Chen の存在従属クラス図と等価であり，併せて，椿・渡辺による ER 図設計手法の正当性を理論的に傍証する．本提案によれば，存在従属クラス図の作成ガイドラインとして，1) 存在従属クラス図の独立クラスと従属クラスを分けて認識，2) オブジェクトのライフタイムに着目して，従属クラスを同定し，3) 特定タイミングのデータ状態を記録する「タイミング型」と，ある時間的期間におけるオブジェクトの存在を表現する「期間型」の 2 種類の従属クラスを設けた設計を行う，が示される．

キーワード：クラス図，関連，関数従属，ライフタイム，存在従属，リソースクラス，イベントクラス，ER 図

Understanding of Class Diagram based on Object Lifetime

Shigeo KANEDA[1,a)]　Akio Ida[1]

Abstract: UML specification categorizes “association” as a semantic relationship. This definition, however, is not a practical guideline for software development. To resolve this problem, this paper stands on the position that association is equal to the functional dependency of relational database. The association that one of the two multiplicities is equal to one represents functional dependency. On the other hand, the multiplicity=0 is often required in a class diagram. The multiplicity=0 can be expressed by using a class having timestamp (the creation time), or a class having lifetime (between the creation time and the deletion time). Also, this paper shows that the link of association starts from an object having shorter lifetime and the short lifetime of the object should be included by the longer lifetime of the linked object. The derived configuration of a class diagram is equal to the Chen’s existence dependency class diagram, and the proposed theory supports the ER diagram design method proposed by Tsubaki and Watanabe. The proposed understanding of a class diagram conducts the following steps for class diagram creation; 1) Concepts of “Independent Class” and “Dependent Class” of the existence dependency class diagram methodology should be adopted, 2) Independent classes and dependent classes are identified by the lifetime analysis for each class. 3) Finally, a timestamp or lifetime (creation time and deletion time) is given to the each dependent class, if it is necessary.

Keywords: Class Diagram, Association, Functional Dependency, Existence Dependency, Resource Class, Event Class

1. はじめに

本論文では，UML 静的モデルの代表的存在であるクラ

[1] 同志社大学大学院理工学研究科
Graduate School of Science and Engineering, Doshisha University, Kyotanabe-city, 610-0321, Japan
[a)] skaneda@mail.doshisha.ac.jp

Vol.2016-DBS-163 No.19
Vol.2016-IFAT-123 No.19
2016/9/15

ス図における「関連 (association)」に着目する．関連に関して，UML の定義 [1] では，以下の様に記述されている．

An association specifies a semantic relationship that can occur between typed instances.

意味的に関係しているところには関連を引くべきとなる．しかし，ソフトウェアの設計方法の指針として見れば，抽象的に「意味的」と言われても，具体的にどの様に設計したら良いか分かり難い．

そこで，技術的にわかりやすい定義を導入するべく，本論文では，**「関連（association）は関係データベース（relational database）における関数従属である」**との立場に立脚する．すなわち，「クラスに所属する一つのインスタンスが決まれば，関連先クラスのインスタンスが一つ決まる」のが関連であるとする[*1]．ソフトウェア要素的には，関数従属の関連は，一方向性のポインタである．

もともと，関数従属と関連の関連性は強い．一方の側の多重度が 1（そして，1 に限る）関連は，関数従属と等価である[*2]．本論文では，更に，関連が張られている 2 つのオブジェクト間のライフタイムの関係を分析する．この分析から，**関数従属の関連は，ライフタイムの短いオブジェクトから，ライフタイムの長いオブジェクトに向けてのみリンクできる**ことが証明される．そして，実際のクラス図で必要となる多重度 0 も，オブジェクトにタイムスタンプを持たせることにより，表現できる．

以上の理論的分析により得られたクラス図の構成は，Chen[2] により提案され，本論文の著者（井田）が研究を進めている「存在従属クラス図」[3] そのものである．更に，本論文の分析結果は，実務家の間でよく知られた，椿・渡辺による ER 図の設計法 [4][5] に対しても，理論的裏付けを提供する．

以下，第 2 章では，関数従属の関連について考察する．第 3 章では，属性と関数従属について考察する．第 4 章は，本論文の中心であり，関連とライフタイムの関係を分析する．第 5 章では，本提案の手法の適用例として，実務家の間で知られている「花束問題」の仕様書からクラス図を作る例を示す．第 6 章は，本論文のまとめである．

2. 関数従属とクラス図

関係データベースにおける関数従属の概念は良く知られており，以下の関数式で表現される．ここで，$Func$ は関数を示し，ドメイン X の要素を X_i，それに対するドメイン Y の要素を Y_i とする．その意味は，「ドメイン X の任意の X_i に対して，ドメイン Y 中の Y_i が一意に決まる」である．

$$Y_i = Func(X_i) \tag{1}$$

図 1 には，関数従属の図形イメージを示す．この関数従属では，以下の性質 1 がある．

【性質 1：関数従属の前提】

性質 1-1　2 個以上の相異なる X_i に対して，同一の Y_i が出力される場合がある．

性質 1-2　ドメイン X，ドメイン Y の全ての要素を対象としている．例えば，ドメイン Y には，ドメイン X の要素から指定できない要素は想定されていない．

性質 1-3　時間軸（ライフタイム）は考えていない．時間に無関係に関数は成立する．

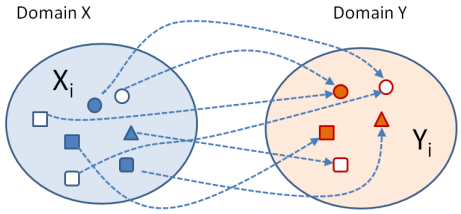

図 1　関数従属
Fig. 1　Functional Dependency

上記の関数従属の定義では，実際にリレーション上に値となって現出しているかどうかは別問題である．現実世界への現出は，各関数従属ペア独自に独立して発生し得る．RDB で言えば，プライマリーキー（本項の議論では属性 1 個から成るものとする）とそこから関数従属関係にある属性 1 個の，合計 2 属性から構成されるタプル（レコード）が，他とは独立に，現実化したり消えたりする．

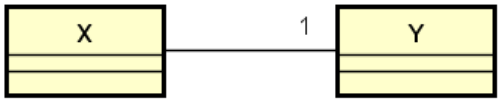

図 2　関数従属のクラス図的理解
Fig. 2　Understanding of Functional Dependency

図 1 は素直に，UML のオブジェクト図とも理解できる．即ち，関数従属が成立している図 1 をクラス図で描くと，図 2 の様になる．多重度はクラス Y 側しか書かれていない．本論文では，多対多の関係は 1 対多に変換されて，存

[*1] 本論文では，UML で言う，クラスの要素である「オブジェクト」には原則「インスタンス」を用いる．オブジェクト指向全般の考え方などを論じる際に，「オブジェクト」を利用したいからである．但し，対象がインスタンスであっても，インスタンスのオブジェクト指向的なとらえ方を論じる場合には，誤解を招かない範囲で，オブジェクトを用いることがある．

[*2] 本論文では，この多重度 1（そして，それに限る）関連を，「関数従属」と呼称する．そこには，「関数従属」と呼ぶ以上，オブジェクト間の関連についても，関連のリンクが走るのは，非推移的な関数従属のみに限定するべきとの発想がある．目的は，関係データベースの第 3 正規系と同様に，生成されたデータ構造のモジュラリティの確保である．

Vol.2016-DBS-163 No.19
Vol.2016-IFAT-123 No.19
2016/9/15

在しないものとする[*3]．多重度=1 は，クラス X のインスタンスが決まれば，クラス Y 側のインスタンスが，常にユニークに決定されることを示す．本稿では，「多重度=1 は関数従属を意味し，多重度=1 のクラスが出力側，関連で接続されたもう一方のクラスが入力側」との前提で議論を進める．

3. 属性値と関数従属性

クラス図の属性を取捨選択する際に，ライフタイムに着目するべきことは，従来から明らかにされてきている [6]．「それぞれの属性のライフタイムが一致しているのであれば，まとめて扱い，クラスのインスタンスとして，一括して追加・消去すべき」となる．言い換えると，前節の関数従属では現出化の単位は，プライマリーキーと属性のペアであったものが，ライフタイムが同じものは，一括して，現出化・消去している．これを，オブジェクトと呼ぶ．結果として，以下の性質がクラスの属性には存在する．

【性質 2：属性と関数従属】

性質 2-1 オブジェクトの識別子（例えば，オブジェクト ID）を含めて，全属性値は，当該オブジェクトが生まれるとともに値を持ち，オブジェクトが存在する限り値を有し，オブジェクトが消滅すると同時に値を持たなくなる．属性値は，このライフタイムの間においてのみ意味を持つ．

性質 2-2 属性におけるライフタイムの一致は，必ずしも，オブジェクト識別子（例えば，オブジェクト ID）に対して，全属性値が，非推移的な関数従属であることを保証するものではない．

性質 2-3 属性 A,B,C の間に，$A \rightarrow B \rightarrow C$ の推移的関数従属があるとする．この場合，A と C のライフタイムが一致して（すなわち，同一のオブジェクトに保存され），B がそれとは異なるライフタイム（異なるオブジェクトに保存されている）になる様な推移的関数従属は考えにくい．結果的に，他に同一のライフタイムを持つオブジェクトが分析対象とする世界に存在しなければ，オブジェクトの属性を，それ以上追加する必要はないと推定される．

上記性質 2-3 を説明する図を図 3 に示す．属性 B のライフタイムが，他の属性 A,C とは異なる状況で，属性 $A \rightarrow$ 属性 $B \rightarrow$ 属性 C と言った推移的な関数従属が現れる可能性は考えにくい．そこで，以下の性質 3 が成立する．

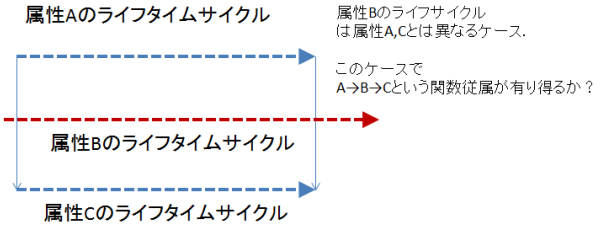

図 3 推移的関数従属が不自然となるケース
Fig. 3 Illegal Transitive Functional Dependency

【性質 3：属性の帰属先の決定方法】

オブジェクト識別子（例えばオブジェクト ID）が与えられた時，オブジェクト識別子と同一のライフタイムをもつ属性は，漏れなく[*4]クラス図に表現せねばならない．収集された属性群とオブジェクト識別子との間で，非推移的関数従属関係を確認する必要がある．

4. ライフタイムから見たクラス図の構成

以上の準備のもとで，「関数従属が関連である」との立場から，オブジェクト（インスタンス）のライフタイムと関連の関係について，考察する．これにより，関数従属を関連とするクラス図が持つべき姿を明確化する．関連のインスタンスであるリンクは，2 つのオブジェクト間に張られるものである．したがって，本章では，2 つのオブジェクトのライフタイムの相互関係に着目する．

4.1 ライフタイム

図 4 は，2 つのオブジェクト間 (A,B 間，および C,D 間) のライフタイムの関係を示す．2 つのオブジェクトのライフタイムが完全に一致するなら，前章の議論から，一つのオブジェクトに属性として統合されるべきである．従って，2 つのオブジェクトのライフタイムの包含関係は，図 4 に示すように，1) 一方が他方を包含する「包含関係 (A と B)」，及び，2) 2 つがすれ違い，共通部分は存在するが，それぞれのオブジェクトが孤立して存在する時間がある「すれ違い関係 (C と D)」，の 2 通りしか有り得ない．2 つのオブジェクトのライフタイムが全くオーバーラップしないなら，そもそも，リンクは引けないので，その様な，ライフタイムのケースは除外される．

4.2 包含関係のライフタイム

本節では，2 つのオブジェクトが包含関係をなす場合を分析する．図 4 においても分かる様に，一方のライフタイ

[*3] 本論文で取り上げてゆく存在従属クラス図では，すべのクラスに独自のオブジェクト ID を振る様なことはせずに，例えば，存在従属の上流側クラスのプライマリーキーを下流側のクラスが引き継ぐ形の，ER 図に近いクラス図のみを扱っている．操作（メソッド）は登場しない．したがって，本論文で扱うクラスは，そのまま ER 図として DB 設計が可能となる．本論文で「クラス図」と呼ぶものは，「ER 図」と理解して頂いても問題ない．

[*4] 「もれなく」は理論的には期待されても，現実的には不可能と考えられる．例えばある従業員オブジェクトと同一のライフサイクルタイムを持つ属性は無数にある．現実には，「業務要件に照らして漏れなく」あたりに落ち着くものと思われる．

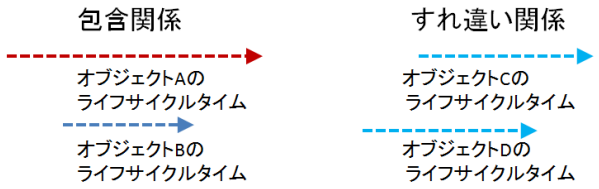

図 4 ライフタイムの包含関係
Fig. 4 Relations between Lifecycles

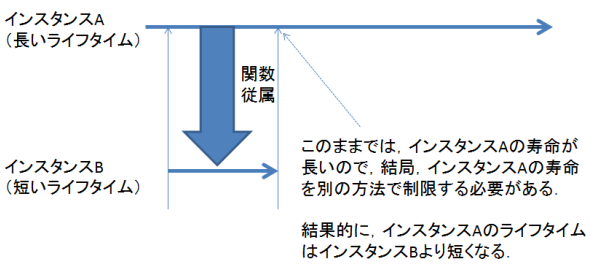

図 5 包含関係のライフタイムと関連（1）
Fig. 5 Relations between Lifecycles

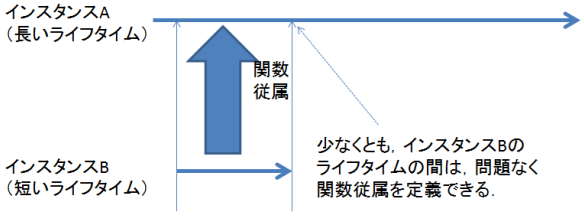

図 6 包含関係のライフタイムと関連（2）
Fig. 6 Relations between Lifecycles

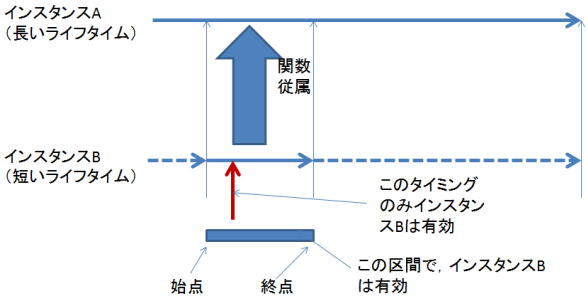

図 7 多重度=1 への変換
Fig. 7 Conversion for Multiplicity=1

ムは完全に他方を包含する．まず，関数従属について，図 5 に示す様に，ライフタイムが長い側が，入力側クラスであったと仮定しよう．この場合，関数の入力値を担当するクラスの方が寿命が長く，出力値を定義できない入力値が必然的に存在する．これは矛盾であり，ライフタイムが長いオブジェクトから，ライフタイムが短いオブジェクトへの関数従属の関連は，存在し得ない[*5]．

次に，反対に，ライフタイムの短いオブジェジェクトから，ライフタイムの長いオブジェクトに関数従属の関連を引く場合について考ええる．この場合，関数の定義域の問題は生じない．

以上から，「クラス間のライフタイムが包含関係にある際には，かならず，ライフタイムが短い側からライフタイムの長い側に関数従属関係のある関連が成立する．この様子を図 6 に示す．

ただし，このままでは，ライフタイムの長い側の関連の多重度は 1 ではなく，「0..1」である場合が想定される．ライフタイムの関係で，リンクが張れない期間があるからである．多重度 0 を含んでは，関数従属にならない．そのためには，多重度「0..1」を「1」に変える必要がある．しかし，これは，ライフタイムが短かかった側のオブジェクトのライフタイムを引き延ばして，長い側に一致させることを意味する．双方のオブジェクトがシステムの上では，ずっと存在しているイメージである．これでは，もともとのオブジェクトの仕様を満足しない．この状態を解決するためには，図 7 の様に，ライフタイムの短い側のオブジェクトに，以下の 2 つの属性のどちらかを与えて[*6]，アプリケーション側でソフトウェア的に対応する必要がある．

タイミング型 有る瞬間のデータ状態を記憶するクラスである．たとえば，「発注」「受領」などである．あるタイミングにおける現実社会のデータ状態を写し取ったものである．仮に，このタイミング型は 1 個しかインスタンスが無いなら，上流側の包含するクラスが決まればユニークに決まる．しかし，何度も同様の行為が繰り返される場合には，タイムスタンプ，シーケンス番号等を追加して，インスタンスを識別する必要がある．

期間型 たとえば，マンションへの入居の様に，開始時点と終了時点があるクラスである．この場合も，何度もインスタンスが同じ上流側クラスから生成される場合を識別するためには，タイムスタンプやシーケンス番号を用いて識別する必要がある．期間を持つから，有効期限の開始から終了までを，インスタンスの属性として持たせる必要がある．

以上のタイミング型と期間型の導入と区別は，現場のプログラマがしばしば行っている実装手法である．ここで，注意が必要なのは，この 2 種類があれば，ライフタイムを表現するに必要かつ十分であり，結果として，上流側のクラスの多重度は 1 となって，関数従属性が担保されるこ

*5 無理に，ライフタイムの長い方から，短い方に関数従属の関連を引くと，上述のイベントクラスのオブジェクトの様に，ライフタイムが長い側のある一定の期間のみをソフトウェア的に有効として，データ構造を扱う必要がある．この場合の「一定の期間」は，当然，ライフタイムの短かった側のオブジェクトの存在期間を超えるものではなく，包含されている必要がある．このような便宜的処置を施しても，結局，関数従属性は，ライフタイムの短い側を入力としたものと同一になる．

*6 上流側のクラスが全て独立クラス（リソースクラス）である場合，このライフタイムの限定に関する情報を付加する必要がないケースがある．

とである．いわゆる「もの–こと–もの」パタンは，このタイミング型である．しかし，「こと」クラスを作ったのみでは不十分であり，属性としてタイミング情報を付与する必要がある．上記の議論の重要なポイントは，関連に対して，「このインスタンスがきまれば，このインスタンスが指定される」という関数従属の考え方を導入する限りにおいて，包含関係のインスタンス間が持つリンクに由来する関連は，一方の多重度が 1 に限定された，(RDB の) 関数従属のみで十分であると言うことである[*7]．

4.3 すれ違い関係における関連

残された問題は，図 4 右側に示した，「すれ違い関係」における「関連」の姿である．この場合は，2 つのドメイン X, Y の定義域がすれちがっているため，どちらかのライフタイムに一致したドメインを設定できない．つまり，当該クラスに属するインスタンスが，観測時刻によって，関連のリンクになっているのか，なっていないのか分からない．多重度=0 を導入せざるを得ない．以下の性質 4 が成立する．

【性質 4：すれ違い関係の場合】

性質 4-1 すれ違い関係では，1 対多の関連は，2 つのクラスの中の，どちらを出発点としても問題ない．

性質 4-2 すれ違い関係では，通常は，関連が持つ 2 つの多重度に，それぞれ 0 を含むものと推定される．片方のみに 0 を許すと，2 つのクラスのライフタイムの中で，インスタンスが存在しないのに，多重度が 1 以上の期間が現れてしまう恐れがある．

性質 4-2 但し，入力側のオブジェクトに，タイミングや期間の概念（属性）を入れて，ソフトウェア的にライフタイムを制御した場合，多重度 1 が受容される可能性は残されている．

上記の様な，すれ違い関係が発生するのは，お互いに他のオブジェクトの存在に影響されないクラス相互の関連であると思われる．すなわち，椿の言う「リソースクラス」[4]，存在従属における「独立クラス」[3] において，現れる可能性が高い[*8]．しかし，リソースクラス/独立クラスは意味的に相互の距離は遠く，このようなすれ違い関係に関連を引く機会は少ないのではないかと想定される．

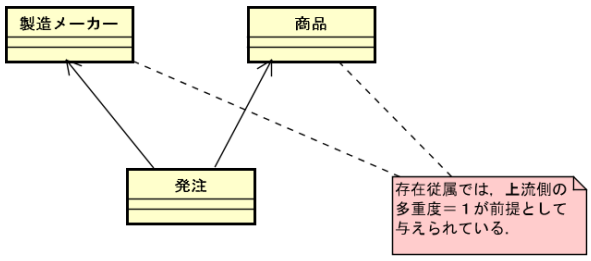

図 8 存在従属の簡単な例
Fig. 8 Existence Dependency

4.4 関連研究

本研究の関連研究としては，2 つある．ひとつは，ER 図の提案者である Chen による「存在従属クラス図 (ER 図)」である [2]．存在従属は，Chen により提案されたにもかかわらず，なぜか，スリー・アミーゴには注目されず，UML の規格には取り込まれなかった．井田は，この存在従属について，重要性を指摘している [3]．図 8 は，存在従属クラス図（井田の記法による）の簡単な例である．上流側の 2 つのクラスに，下流側のクラスから，矢印の関連が伸びている．矢印の受け入れ側が，前提となるクラス（群）であり，図 8 の例では製造メーカーと商品（の種類を表すクラスである）のインスタンスが決まらないと発注ができない．いわゆる「もの–こと–もの」パタンの例である．

存在従属では，1) 他のクラスの存在とは無関係に独立して，生成されたり，消去されるクラス，2) 他のクラスのインスタンスの存在を前提として生成可能なクラスに分けて考える．前者を「独立クラス」，後者を「従属クラス」と呼んでいる．図 8 の例では，商品や製造メーカーのインスタンスは，他のクラスのインスタンスの状態に無関係に，生成できる．これが独立クラスである（椿はリソースクラスと呼んでいる [4]）．それに対して，従属クラスは，他のクラスのインスタンスの存在を前提として，生成される．「発注」はその例である．. 発注行為は，商品のインスタンスと製造メーカーのインスタンスが存在しなければ，生成できない．従属クラスを，椿は，「イベントクラス」と呼んでいる [4]．

存在従属では，その存在を前提とする側のクラスが持っている多重度は 1 である．明らかに，関連は，関数従属である．しかし，存在従属の議論では，何故に，その様な制約された技法で，自由にドメインを表現できるかについては，触れられていない．ライフタイムについても，ほとんど言及は見られない．しかし，本論文の分析によって，存在従属の枠組みは網羅性があることがわかる．存在従属で

[*7] タイミング型のインスタンスでは，生成タイミングで，関連した周囲のインスタンスをたどり，その瞬間のデータ状態を，ビューの様に記録することがしばしば行われる．この理由も自明である．タイミング型は，特定タイミングのデータ状態を記録するためのインスタンスであるから，生成タイミングで，推移的関数従属により辿ることのできるデータ値を（後で必要なものについて）保管していることになる．このビュー的なデータは，第 3 正規系ではないので，以後の時刻におけるデータ内容管理は，プログラマの責任となる．これに対して，期間型では，時間が間隔を持っているので，このアプローチは使えない．

[*8] いずれも，他のオブジェクトの存在に無関係に，オブジェクトが生成されるクラスである

は，下流側の従属クラスから上流の独立クラスの側へ向かって従属関係が伸びているが，ライフタイムで決定しているものであり，逆方向への関連はあり得ないことを，本論文は示している．

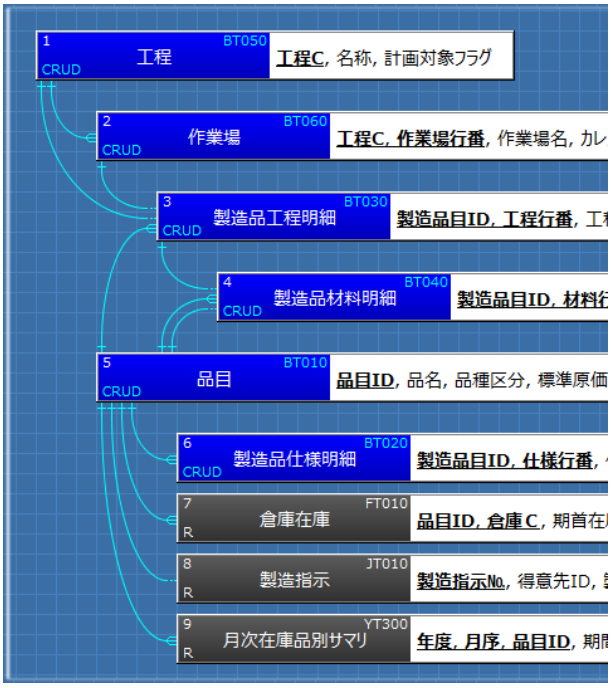

図 9 渡辺による ERD の例
Fig. 9 An Example of Watanabe's ERD

もうひとつの関連研究は，渡辺の業務システムの設計法である [5]．図 9 は，渡辺のアプローチでの設計を支援するツールである X-TEA に付属していたサンプルの一部である．左側へゆくほど，上位のクラスとなるが，上位のクラスに延ばされた関連の多重度が軒並み，1 であることがわかる．これは，一見すると，一種の設計上の制約であり，業務システムだけがこの様になると思いがちである．しかし，本論文の意味するところから，この方法論には汎用性があることがわかる．ただし，タイミング型や期間型のテーブル（クラス）を導入して，対策をする必要があることを，本論文は示唆する．

渡辺は，関連として，参照関係，親子関係，派生関係の 3 種類の関連で，データベースは設計できるとする[*9]．これも，なぜ，それで表現できるかについては言及しておらず，むしろ，結果として得られるプライマリーキー属性の内容によって，関連を区別している．親子関係では名前のとおり，親のクラスのプライマリーキーを子のプライマリーキーに取り込んでいる．存在従属と等価である．更に，この構造であれば，子のプライマリーキーから，親を一意に指定できるので，本論文の関数従属性そのものである．つまり，渡辺は陽には主張していないが，渡辺の方法論は，ライフタイムの包含関係を含意しており，存在従属とも等価である．

尚，派生関係は is-a 関係である．is-a 関係は，関数従属とは異なるものであるので，本論文の議論の範囲ではない．しかし，is-a 関係は，上位クラスのプライマリーキー属性（群）と同一の属性（群）を下流側のプライマリーキー属性（群）として利用することはよく知られており，本論文の議論との不整合はない．is-a 関係は，従来研究との差異というわけではなく，本論文の枠組みに追加可能である．参照関係は，一方のプライマリーキーが，他方の外部キーとなっているケースとしている．このケースは，本論文における「すれ違い関係」において現れる，多重度 0 を持っている可能性のある関連である．

本論文が提示しているのは，新しい構成や新しい機能ではない．しかし，従来は提示されていなかった，「どうして存在従属や渡辺の方法論で，すべてのドメインのクラス図が描けるのか？」という保証を与える．その結果，タイミング型や期間型のクラスを（椿の言い方では，イベントクラスのクラスに）設ける必要があることを示している．オブジェクトに，有効期間開始タイミングと有効期間終了タイミングを属性として与えることは，プログラマのレベルで日常行われている．しかし，その背後には，オブジェクトのライフタイムを事実上伸ばして，ライフタイムを無視している RDB の理論の枠内で，オブジェクトを扱おうとする論理が存在する．ライフタイムを無視した設定をしないと，クラス図と RDB とのマッピングも困難となるからであろう．

5. 「花束問題」への適用

本章では，データベース技術者の間で知られた「花束問題（V1.2)」を例として，ライフタイムに着目した分析を行う．花束問題は，佐野初夫により，IT 勉強宴会 (http://www.benkyoenkai.org) の例題として作成されたものである．無料で自由に再配布できるが，再配布の条件が文書全体なので，本論文では引用を差し控える．文書は，以下の URL から取得できる．
http://www.benkyoenkai.org/2015/01/v12.html

上記文書の「文書化されているユーザ要件」の中で，「商品と在庫」，「得意先と受注・出荷」，「発注と入荷」，「代金の扱い」の部分から概念クラス図の作成を行った．帳票や画面の仕様は無視した．

本論文の分析結果から，クラス図の作成に際しては，独立クラス，従属クラスの区別を明確にしつつ，従属クラスについては，タイミング型か期間型かを常に意識する必要がある．図 10 は，花束問題の分析に際して作成したメモ

[*9] 正確には渡辺は J. Martin の IE 表記を応用しており [7]，親子関係，参照関係，派生関係のモデル要素は IDEF1X や IE 表記にも依存型リレーションシップ，非依存型リレーションシップ，および，サブタイプとしてそのまま登場する [8][9].

仕入れ先		独立クラス	
単品	単位本数，有効期限	独立クラス	
提供		従属クラス	期間形
発注		従属クラス	タイミング型
入荷		従属クラス	タイミング型
在庫	（毎日の記録を残す）	従属クラス	タイミング型
商品		独立クラス	
結束		従属クラス	期間形
得意先		独立クラス	
受注	お届け日，届け先，出荷日，お届けメッセージ，お届け先電話番号	従属クラス	タイミング型
変更届け	（受注明細の期間化で対応）	従属クラス	期間形
出荷		従属クラス	タイミング型

図 10 花束問題の分析
Fig. 10 Analysis of HANATABA Problem

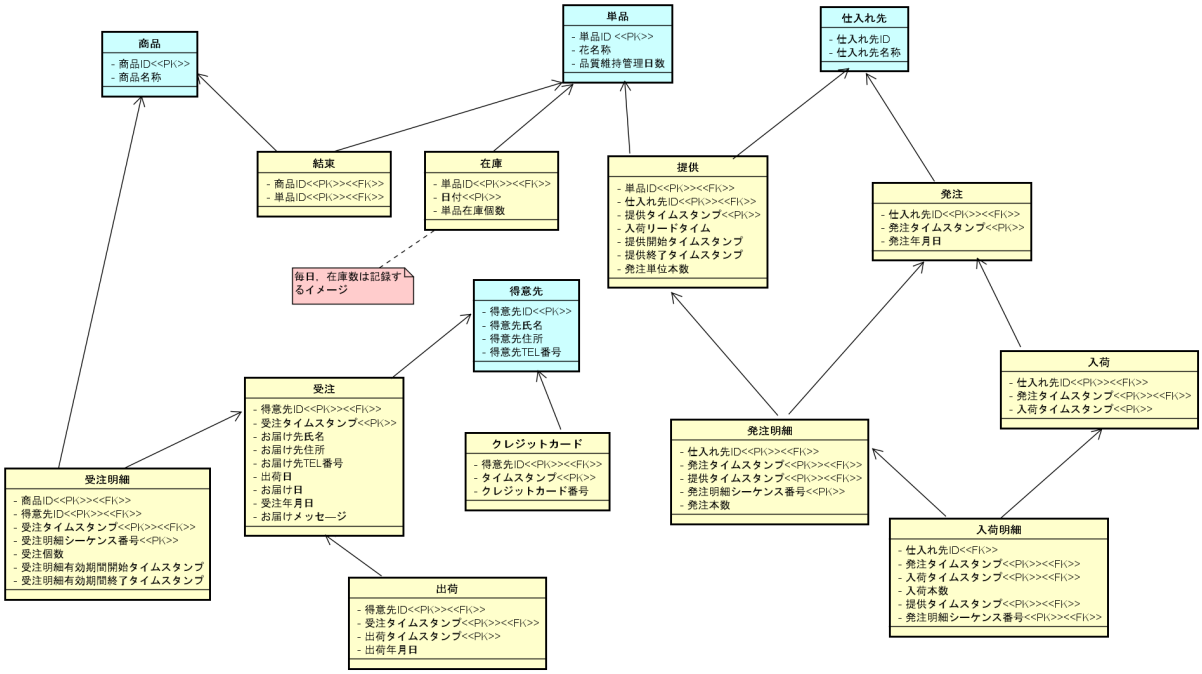

図 11 花束問題のクラス図
Fig. 11 Class Diagram of HANATABA Problem

である．各クラスの候補に対して，ライフタイムを描き，どれに包含されるかを考察する．以前は何となくつけていた，「有効期間の開始日時」「有効期間の終了日時」といった属性も，必要性について自信をもって，付与・設計できる．

図 11 は，実際に生成された存在従属クラス図である．独立クラスと従属クラスは明確に意識されており，従属クラスのプライマリーキー属性（群）は渡辺の設計法を取り入れて，存在従属の上流側のクラスが持っているプライマリーキーはそのまま継承する．なお，仕様書には，発注明細の修正に対応できることとあった．もともと，発注はタイミング型の傾向が強い．しかし，修正が入るのであれば，期間型のクラスとして，修正クラスのような帳票イメージは避けている．本論文の分析結果がでるまでは，上流から多重度 1 で流れてくる存在従属クラス図では，表現できないビジネス内容があるのではないか？といった不安を持つこともあったが，本論文の分析から，存在従属の考え方を素直に適用できることに確信をもって，クラス化の作業が可能となる．

6. 終わりに

本論文では，「クラス図の『関連』は関数従属である」との立場から，オブジェクトのライフタイムに着目して考察する新しいクラス図理解のアプローチを提示した．ただし，本論文の議論では，クラス図には多対多の関係がないことを前提としている[*10]．関連が関数従属であるならば，少なくとも，関連の一方の側の多重度は 1（そして 1 に限る）でなければならない．一般的には，多重度は「0..1」となる可能性があり，多重度 0 の場合は関数従属ではない．しかし，この「0」が導入される理由を考えると，関連の元であるリンクを張る 2 つのオブジェクトが実際に現出して

[*10] 多対多の関連は，交差クラスを導入することで，容易に，1 対多に変換できるので，特に問題ないと考える．そもそも，多対多では，RDB への変換が難しい

いるライムタイムの相互関係が関与している．そこで，クラス図を理解するために，（RDB では顧みられていない）ライフタイムの観点を導入した．その結果，関数従属の入力側のクラスに，1) 特定タイミングで生成されたことを記録するためオブジェクトであることを示す「タイムスタンプ」を属性として持たせる[*11]，2) 一定の時間的な期間のみオブジェクトが有効であることを示す，有効期間の開始日時と，終了日時を属性として持たせる[*12]，の 2 つのタイプのクラスを設けることによって，多重度 1 の関連の実現が担保できることを示した．そして，ライフタイムが短い側のオブジェクトのライフタイムが，ライフタイムが長い方のオブジェクトのライフタイムに包含されるケースにおいて，関数従属関係は，ライフタイムが短い側のオブジェクトから，長い側のオブジェクトに関数関係が及ぶ場合に限定される．

本論文のライフタイムに着目した関連の理解は，新しい機能をクラス図に与えるものではない．しかし，その理解によって，以下の点が明らかとなっている．

- Chen の存在従属クラス図（ER 図），椿・渡辺の DB 設計のアプローチは，上流側のクラス（テーブル）への関連の多重度を 1 にしており，一見すると，表現能力が限定されている様に見える．しかし，このアプローチに理論的妥当性があり，これによってモデルの表現能力の網羅性は担保されている．
- 上記の Chen の存在従属，あるいは，渡辺の設計法では，上流側への関連は，多重度=1 である．従って，「関連は関数従属性である」と理解できると考える．具体的に，「このクラスのインスタンスが決まれば，それによって関数関係にある別のクラスのインスタンスが指定される」ことを意味する．ソフトウェア的には一種のポインタを意味する．
- 多重度 1 の実現は，1) 特定タイミングに有効であることを属性で表現するタイミング型のクラス，又は，2) オブジェクトが一定期間に有効であることを属性で示した期間型のクラス，の 2 つのクラスを設けることにより，必要十分条件で担保される．

以上の理論的分析から，クラス図を存在従属，あるいは，椿・渡辺の設計法で作成する場合，オブジェクトのライフタイムの関係を重視することが示唆される．また，上記のタイミング型のクラスとは，「もの–こと–ものパタン」の「こと」クラスを意味している．しかし，関数従属性を担保するには，タイミング型のみでは不十分であり，期間型クラスの導入が必要，かつ，十分である．この様な，期間型の導入は，プログラマは日常的にやっていることであり，それ自体に新規性はない．むしろ，本論文で見えてきたものは，タイミング型・期間型クラスの導入は，クラス間の関連を関数従属性に限定できることを意味していること，結果的に，井田や渡辺が提示しているように，上流側のクラスのプライマリーキー属性を下流側（渡辺の表現では，親子関係の関連の下流側）のプライマリーキー属性に組み込むことへの理論的保証である．逆に言えば，上流側の識別子をプライマリーキーとして下流側のクラス（テーブル）に持ち込むことは，外部キーを利用する型で，ライフタイムの包含関係を，DBMS によるインテグリティ検査が自動的に可能となる．本論文の分析結果から見ても，クラス毎に無条件にオブジェクト ID を与えるオブジェクト指向分析のアプローチが妥当なものかどうかは疑問が残る．このような ID の付与方式では，いつの間にかライフタイムの視点が忘却されてしまい，インスタンス間の存在に関する制約も組込めないからである．

[*11] 本論文では，「タイミング型」と呼んでいる．いわゆる「もの–こと–もの」パタンの「こと」クラスである．
[*12] 本論文では，「期間型」と呼んでいる．

参考文献

[1] OMG Management Group: *OMG Unified Modeking Language^{TM} (OMG UML), Superssructure Version 2.4.1"*, p.36, OMG Management Group (2011)
[2] P. Chen : *The Entity Relationship Approach to logical Database Design, QED*, Information Science, Wellesley (1979)
[3] 井田明男，金田重郎，熊谷聡志，藤本明莉：存在従属性に着目した論理要件ロバストなドメインモデルの作成- ドメインクラス図をユビキタス言語として用いるために-，情報処理学会論文誌，Vol.56, No.5, pp.1340-1350 (2015)
[4] 椿正明：名人椿正明が教えるデータモデリングの技，翔泳社（2005）.
[5] 渡辺幸三：販売管理で学ぶモデリング講座，翔泳社（2008）.
[6] 金田重郎，井田明男，酒井孝真，熊谷聡志：日本語仕様文からの概念モデリングガイドライ－行為文と関数従属性に基づくクラス図の作成－，電子情報通信学会論文誌 D，Vol.J98-D，No.7，pp.1068-1082 (2015)
[7] 渡辺幸三：業務別データベース設計のためのデータモデリング入門，日本実業出版社（2001）
[8] J. Martin : *Information Engineering Planning & Analysis*, Book II., Prentice-Hall, Englewood Cliffs, NJ (1990)
[9] I. Yeol,M.Evans: *A Comparative Analysis of Entity-Relationship Diagrams*, Journal of Computer and Software Engineering, Vol.3, No.4, pp. 427-459 (1995)