［研究論文］

関連を１対多関連のみに限定した「工場モデルクラス図」とその汎化関係への拡張

“Factory Model Class Diagram” using one-to-many Associations only and its Extension to Generalization Relationships

金田 重郎
Shigeo KANEDA

同志社大学　名誉教授
Professor Emeritus at Doshisha University

要旨

UML クラス図は，概念データモデリングにおける重要な分析ツールである．著者らは，既に，1 対多関連のみを用いるクラス図は，クラスをノード，1 対多関連をアークとすると，「有向非巡回グラフ（DAG, Directed Acyclic Graph）の推移簡約」に等しいことを示している．この種のクラス図では，時間的に古いものから順番に，処理単位である業務処理プロセスの出力結果に対応するクラスを，次々とクラス図に追加する形で，見通し良くモデルを構築できる．本論文では，この種のクラス図を工場モデルクラス図と呼ぶ．しかし，既存研究では，1) 関係モデルを証明に利用している等，証明が複雑で全体像把握が困難，2) 工場モデルをヒューリスティックスと誤解する恐れ，3) クラス図における重要な「関係」である汎化の除外，等の課題があった．これら問題を解決するため，本論文では，1 対多関連のみを利用するクラス図が，DAG の推移簡約となることを，オブジェクト指向を出発点として証明する．本論文における証明は，論理の展開が既存研究より簡明であり，工場モデルクラス図が前提としている思想を理解し易いと考える．汎化関係については，工場モデルクラス図のクラスの位置に，スーパークラスとサブクラスをひとまとまりに置く手法が妥当と判断する．

Abstract

UML class diagrams are important tools in Conceptual Data Modeling. The authors have already proved that a class diagram using one-to-many associations only is a transitive reduction of DAG (Directed Acyclic Graph), by considering classes as “nodes” and one-to-many associations as “arcs”. This type of class diagram has a processing structure like a product processing and assembly factory. For this reason, the class diagram is called “Factory Model Class Diagram”. In Factory Model Class Diagram, it is possible to model the target business as the sequence of business steps in chronological order. However, in the existing research, 1) the proof is complicated and hard to understand since the use of relational model, 2) there is a risk of misunderstanding Factory Model Class Diagram as heuristics. In addition, 3) “generalization relationships” are excluded. Therefore, in order to solve these problems, we prove from an object-oriented viewpoint that a class diagram using one-to-many relationships only is a transitive reduction of DAG. The proposed proof is simple and easy-understanding. Regarding generalization relationships, the superclass and its subclasses should be treated together as one class of Factory Model Class Diagram.

1. はじめに

UML クラス図(Class Diagrams)は，概念データモデリング（CDM, Conceptual Data Modeling）における重要な分析ツールである．著者らは，1 対多関連のみを用いるクラス図は，クラスをノード，関連をアークと見なすと，「有向非巡回グラフ（DAG, Directed Acyclic Graph）の推移簡約」となることを既に示している[1]．DAG の推移簡約は，「ハッセ図」と呼ばれることがある．DAG の推移簡約構造を持つクラス図は，「材料を加工して製品を作り上げる加工組立工場」のような処理構造となる．これを，本論文では「**工場モデルクラス図**」と呼称する．尚，「工場モデルクラス図」は，著者らの既存発表[2][3][4][5][6]では，RPDD, EADD と呼称していたものに等しい．アルファベットによる造語では理解し難いので，本論文では，この様に呼称する．また，本論文では，クラス図の操作（メソッド）は扱わない．その意味では，本論文の議論は，ER 図にも成立する．

工場モデルクラス図では，1) 対象ビジネスを構成する単位である業務処理プロセスを，時間的に早く実行されるものから順番に想起し，2) 各業務処理プロセスの実行により得られる結果を保存するクラスを作成し，3) 次々とクラス図に追記して行くことにより，モデル（クラス図）を作成できる[1]．１対

［研究論文］
2022 年 3 月 18 日受付, 2022 年 4 月 30 日受理

多関連の利用により，クラス図の描き方に一定の制約を与え，クラス図の描き方の指針とすることを狙いとしている．尚，1 対多関連のみの限定利用は，クラス図の表現可能範囲を狭めることは無い．

しかし，既存研究は，1) 工場モデルクラス図と関係モデルとの間における記述能力の等価性を証明に用いたために，証明プロセスが複雑化し，2) 工場モデルクラス図の構造を，ヒューリスティックスと誤解しやすい表現となっており，3) 工場モデルクラス図の背後にある設計思想を理解することが困難となっていた．更に，4) 既存研究では，クラス図の「関係」の中でも重要な「汎化関係」をモデルに取り込んでいなかった．

上記課題を解決するため，本論文では，オブジェクト指向の立場から，どのような関連がクラス図には必須かを第一に問題とする．この出発点からスタートして，工場モデルクラス図のトポロジー構造が「DAG の推移簡約（ハッセ図）」に等しいことを，出来るだけ簡明に証明する．また，汎化に関しては，工場モデルクラス図のクラスの位置に，スーパークラスとサブクラスを一体とする，汎化関係を置くべきことを論じる．

以下，第 2 章では議論の背景を述べる．第 3 章では証明の概要を示す．第 4 章，第 5 章では，証明の各ステップを詳しく説明する．第 6 章では工場モデルクラス図への汎化関係の導入について論じる．第 7 章では，工場モデルクラス図では表現できない場合について論じる．第 8 章はまとめである．

2. 議論の背景

UML クラス図は，ソフトウェア開発の種々の工程で利用されるツールである．詳細な議論の前に，いくつかの点を確認しておきたい．

2.1. 概念データモデリングにおける利用

本論文は，ソフトウェア開発における，初期の開発工程，即ち，要求分析・要件定義工程において行われる，CDM (概念データモデリング）への適用を想定して，クラス図作成のための方法論を提示している．ただし，CDM については，種々の立場からの理解・定義がある．本論文では，「分析対象ビジネスに関係した概念，及びそれらの関係性について，クラス図を用いて明確化すること」と理解する．モデリングに際して，UML クラス図以外のシーケンス図等との併用は考えない．ただし，本論文のクラス図と，ユースケース図・シーケンス図等との併用を妨げるものではない．

CDM における記述対象は，バウンダリー，コントロール，エンティティの 3 層中，エンティティ層のオブジェクトである．通常，エンティティ層のオブジェクトは，パーシステントオブジェクトであり，関係モデルデータベース化され，OR マッパー等を用いて実装される．パーシステントオブジェクトであることは，対象ビジネスにおいて管理・保存されるべき，重要な情報であることを意味する．

2.2. モデラーによる「1 対多関連」の多用

クラス図によるモデリングは，初学者にとっては難しい課題である．クラス図の表記法は書籍により紹介されているが，「どうやってクラス図を作成するのか？」が分からないからである[7]．クラス図作成のための方法論が望まれる．

クラス図の作成方法を知るひとつのアプローチは，ベテランモデラーの作成例を観察し，そこから何らかの法則性を見出すことであろう．そこで，渡辺幸三[8]，平澤章[9]のモデリングに関する文献を参照した．これら文献に記載されたクラス図・ER 図（Entity Relationship Diagrams）では，関連として，1 対多関連が多く利用されている．1 対多関連のみで記述されているサンプルも多い．

また，椿正明は，「情報システムは加工組立工場のラインと見なすことができる（文献[10]，P.25）」としている．椿正明の主張は，多くのソフトウェア・エンジニアが感じている点であると思われる．

これらの「一定の傾向」は，モデラーの個人的嗜好に起因するのではなく，クラス図の持つ本質的な法則性の反映である可能性がある．もし，そうであるなら，その本質的な法則性は，クラス図を作成する為のガイドラインとなる筈である．

2.3. DAG である事を目指す理由

クラス図作成のガイドラインの目的は，描きやすく・理解しやすいクラス図の作成方法を提供する事にある．本論文では，クラスをノード，関連をアークと見なすことにより，クラス図が DAG（有向非巡回グラフ）となる様に，関連を 1 対多関連に限定する．DAG 構造を支える半順序関係は，1 対多関連が含意する，時間的前後関係である．

クラス図が DAG であるなら，クラス（ノード）をトポロジカルソート可能である．言い換えると，クラスを，古いものから順に次々とクラス図に追記してゆくだけで，クラス図が完成する．時間的前後関係という理解し易い尺度に従って，クラス図にクラスを次々と追加すれば，クラス図を作成できる．

2.4. 本論文におけるアプローチ

本論文の結論である工場モデルクラス図の構成は, 既存研究[1]と同一である. しかし, 既存研究では, 1) 関係モデルを証明プロセスに持ちこんだため，証明過程が複雑化，2) 1 対多関連が材料-中間製品の関係になることを，根拠なく提示，3）工場モデルクラス図をヒューリスティックスと誤解し易い記述等，工場モデルクラス図が背後に持っている思想を理解することが難しい状況であった.

そこで，本論文では，既存研究とは異なり，「そもそも，クラス図としては，どのような関連が必要なのか」との原点に立ち帰り，オブジェクト指向を起点として，直線的な証明により，「1 対多関連のみから構成されるクラス図は，DAG の推移簡約となる」ことを証明する．証明は，短くはない．しかし，論理は比較的単純であり，工場モデルクラス図の背後にある思想を理解し易いと考える.

尚，UML クラス図には，関連・集約・コンポジション・汎化以外にも，多数の関係性を表現するツールがある．本論文は，工場モデルクラス図を利用すれば，これらの全てを表現できるなどと主張するものではない．本論文が明らかにするのは，工場モデルという基盤になるクラス図の基本構造であり，そこで生成されたクラス図に手を加えて，表現を拡張することは，何ら問題ない.

工場モデルクラス図は，幾つかの側面を持つツールである．証明は長くなるので，次章（第 3 章）においては, まず, 結論である工場モデルクラス図を説明する. 次に, クラス図の正規化について論じる. 第 3 章の最後で，証明の概要を示す．1 対多関連を出発点にして，直線的に，DAG の推移簡約であることが示される．証明の各ステップの詳細は，第 4 章，第 5 章に示す.

3. 証明の概要

3.1. 工場モデルクラス図

本節では, そのトポロジー構造が, DAG の推移簡約である「工場モデルクラス図」について説明する. クラス図は「クラス」と「関連」から構成される．図 1 は，工場モデルクラス図の例である．関連の多重度は,「多重度＝1」側のみを表示している．関連は，すべて 1 対多関連である．本論文では,「1 対多関連」とは，少なくとも一方の多重度が「1」との意味であり,「多」側が，例えば,「0..1」であるようなケースも包含する．図 1 においては，必要に応じて，属性名，関連名，多重度，関連端名等を追記可能である.

本論文の議論では,「関連」に「集約」「コンポジション」を含めて考えている．集約・コンポジションは，関連と同様に，2 個のクラスのインスタンス間のリンクに関する制約である．関連と同様に，集約・コンポジションは，多重度を持っている．親（全体）側のクラスが多重度「1」になり，子（部分）側のクラスが多重度「多」となる．関連との唯一の違いは，親・子のクラスの結びつきが強く，基本的には，親・子のインスタンスが同時に作られるとしている点である．このことは，後述する「1 対多関連が時間的前後関係を持つ」とする議論と関係する．通常，1 対多関連は時間的前後関係を含意する．集約・コンポジションでは，これが同時となる．時間的前後関係に「同時」を加えても，工場モデルクラス図のトポロジカルな構造に影響を与えることはない.

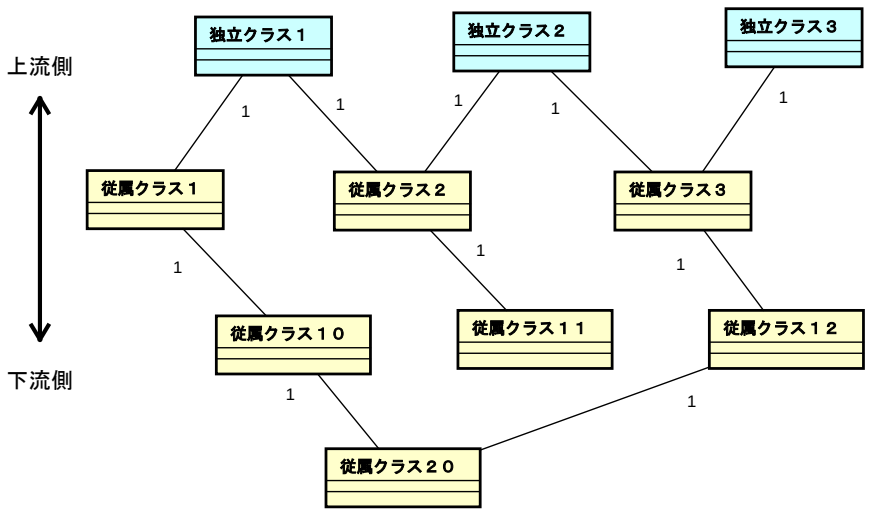


図 1　工場モデルクラス図の作成イメージ

クラス図で２クラス間の「関係」としては，関連・集約・コンポジション以外に，「汎化」が良く知られている．汎化は，関連とは全く意味が異なり，インスタンス間のリンクには無縁である．あくまでもクラス間の関係のみを表現しており，多重度を持たない．このため，関連・集約・コンポジションとは別に議論する（第６章）．

工場モデルクラス図の用語を説明しておく．「多重度＝多」側のクラスから見て，「多重度＝1」側のクラスを，「**上流側**」クラスと呼称する．逆に，「多重度=1」側のクラスから見て，「多重度＝多」側のクラスは，「**下流側**」クラスと呼称される．図１を，工場モデルクラス図と呼んでいる理由は自明と思われる．独立クラスを初期材料として，時間的な推移に従って，次々とクラス作成が積み重なってゆく様子が，あたかも製品の加工組立工場の様に見えるからである．

工場モデルクラス図は，クラスを「**ノード**」，１対多関連を「**アーク**」と見なすと，グラフになる．ここで，アークの方向は，１対多関連の「多」側から「１」側へ向かう矢印で表現する．1 対多関連のみを用いたクラス図（工場モデルクラス図）は「DAG の推移簡約」となる[1]．モデリングに誤りが無ければ，アークがループをなすことはない．「推移簡約」の意味については，次節（3.2 節）で簡単に説明する．

工場モデルクラス図には 2 種類のクラスがある．「**独立クラス**」と「**従属クラス**」である．独立クラスとは，そこから上流には上流側クラスが存在しないクラスである．DAG の性質として，1 個以上の独立クラスが必ず存在する．独立クラスでは，他クラスのインスタンス状態には無関係に，インスタンスを追加できる．一方，従属クラスは，何らかのクラスを上流側に持つクラスである．

従属クラスのインスタンスは，「当該従属クラスのインスタンスとリンクで繋がれるべきインスタンス」が上流側クラスで，事前に準備されていないと，生成できない．つまり，下流側クラスにおけるインスタンスの生成には，上流側クラスに該当するインスタンスが時間的に前に存在していることが必要である．

更に，時間的前後関係が何を意味するかを問題とする．本論文では，「人間（モデラー）は，この時間的前後関係を，『因果関係』と見なしている」と解釈する．因果関係には，「原因―結果」「材料－中間製品」「前提―帰結」等がある．

図 1 に示した様に，工場モデルクラス図では，上部から下部へ向かって，水が流れ落ちているような構造を持つ．この水の流れは，時間的前後関係であり，クラス図の作成時には，上から古い順番にモデリングを行い，クラスを追加して行く．

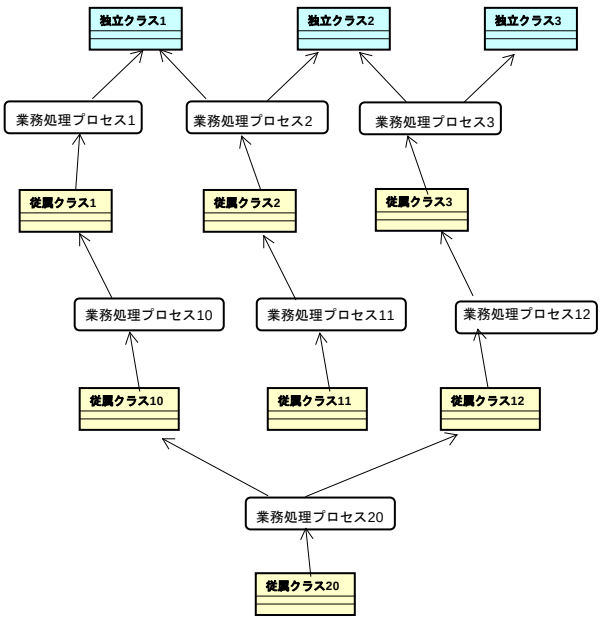


図 2　業務処理プロセスを追記した工場モデルクラス図（テンプレート）

従って，クラス図を描く際には，モデラーは，図 2 のようなテンプレートを思い浮かべることになる．ターゲットビジネスの業務処理は，因果関係の連鎖として表現される．上流側クラスは，下流側クラスにとって「原因／材料／前提」となる．下流側クラスは，業務処理の結果，記録されるべき，「結果／中間製品／帰結」である．それぞれの業務処理プロセスの実行結果を保存するのが，下流側クラスである．

尚，ひとつのクラスが，複数の上流側クラスを持つ場合には，複数の材料を用いて，業務処理プロセスが実行されることになる．

UML 静的モデルのひとつであるクラス図には，対象ビジネスの業務処理プロセスに関する情報は記録されないはずである．しかし，図 2 のテンプレートを見ると，クラス図作成プロセスおいて，業務フローを利用している．対象ドメインをクラス図でモデリングする際に，業務フローを全く知らないでも済む，などということはあり得ない．クラス図を描くには，分析対象ビジネスを構成する単位である業務処理プロセスとその実行順序を十分に理解しておかねばならない．

3.2. 工場モデルクラス図の正規化

ここで，工場モデルクラス図における正規化の必要性について，簡単に触れておく．ソフトウェア開発において，モジュラー化は，最優先事項のひとつである．例えば，Codd の関係モデルでは，関係（テーブル）は，正規化される．これは，テーブルに対する CRUD (Create, Read, Update, and Delete)を，できるだけ，独立して実行することを狙いとする．同様にして，図 1 の工場モデルクラス図においても，各クラスの正規化が要求される．即ち，クラスの属性は，オブジェクト ID に非推移的に関数従属する様にデザインされる必要がある．

しかし，工場モデルクラス図（図 1）には，関数従属性に類似した構成要素がもう一つある． 1 対多関連である． 1 対多関連では，“多”側クラスのインスタンスをひとつ選ぶと，ただちに，“1”側クラスのインスタンスがひとつ決まる．この性質は，関数従属性そのものと言ってよい．

モジュラー化を図るためには，1 対多関連も，非推移的な関数従属でなければならない．つまり，図 1 において，ある 1 対多関連（アーク）がもつクラス間の関係性が，別の複数の 1 対多関連（アーク）をつないで実現されてはならない．複数の 1 対多関連の連鎖により表現される 1 対多関連を残すと，「推移簡約」とはならない．何となく 1 対多関連のみでクラス図を描けば，クラス図が DAG の推移簡約になるわけではない．

推移簡約性は，モデル中に現れた複数の因果関係を合成しても表現できるような因果関係をクラス図に残すことを否定している．クラス図上の因果関係（1 対多関連）は，「対象ドメインの業務処理上の因果関係であり，しかも，これ以上分割すると意味が無くなる，直接的な因果関係」とならねばならない．モデラーは，対象ドメインのビジネスを，そこまで本質を知って，理解する必要がある．

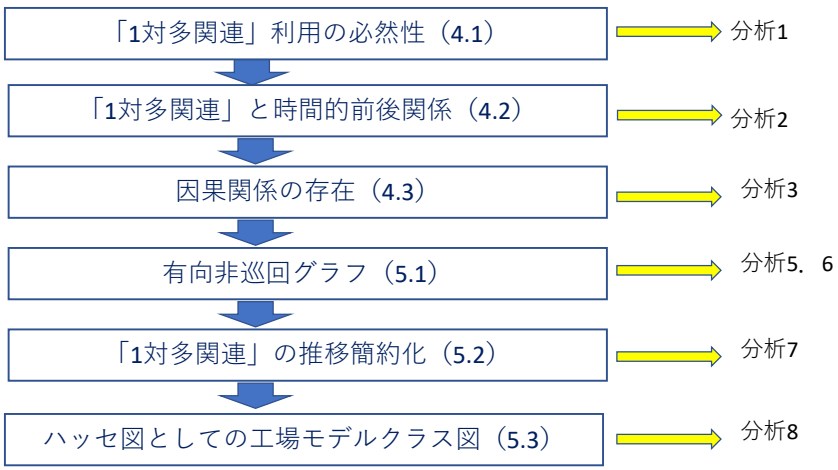


図 3　本論文の証明プロセス[1]

3.3. 本論文における証明プロセス

本章最後に，本論文の証明プロセスの概要を示す．図 3 に，本論文における工場モデルクラス図の導出過程を，節タイトル・節番号を用いて示した．本論文の証明は，図 3 からも分かる様に，1 対多関連を出発点として，一直線の簡明な証明プロセスを持っている．以下，この証明プロセスの概要を説明する．

最初の 4.1 節では，「1 対多関連」がオブジェクト指向モデルと整合性が良いことを確認する．任意のクラス図は，付録 1 に示す様に，容易に，1 対多関連のみを用いたクラス図に変換できるため， 1 対多関連のみの利用は，クラス図で表現される範囲を狭めることはない．

[1] 4. 4 節及び分析 4 は，証明プロセスの外にあるので，この図からは除外している．

次に，1 対多関連が，２個のクラス間の時間的前後関係を表すことを論じる（4.2 節）．更に，時間的前後関係は，因果関係として認知されるとの主張をする（4.3 節）．哲学者ヒュームの主張に基づくものであるが，図 3 の証明プロセスの中では，唯一の人文科学的視点に基づく証明ステップである．以後，数学的に知られた性質を用いて（5.1 節，5.2 節，5.3 節），1 対多関連のみを用いたクラス図は，DAG を推移簡約化したものであることを示す．

本論文の証明プロセスによって，工場モデルクラス図がどのような前提条件の下で成立しているかを理解できる．逆に言えば，前提条件に合致しない対象に対するモデリングは，再検討が要求される．

4. 「１対多関連」の限定利用

本章（第 4 章）は，証明を構成する各ステップの前半である．1) クラス図には 1 対多関連の利用が妥当であること，2) 「1 対多関連」は，クラス間の時間的前後関係を含意すること，そして，3) クラス間の因果関係を示唆することを論じる．

4.1. 「1 対多関連」利用の必然性

本節では，オブジェクト指向における処理に際して，いかなる種類の「関連」が妥当であるのかを分析する．図 4 は，オブジェクト指向における処理の様子である．処理は，各クラスの操作（メソッド）により実現される．処理を実行するクラスは，何らかの方法で，必要な情報を持っているクラス（のインスタンス）から情報を仕入れる必要がある．

図 4 においては，クラス B が処理を実行する．クラス B における処理には，クラス A とクラス C から得た情報が必要である．この場合，クラス B からクラス A へ，及び，クラス B からクラス C へ関連が引かれていれば[2]，クラス B は，クラス A（のインスタンス）及びクラス C（のインスタンス）から情報を取得できる．

図 4 では，処理の実現には，クラス A 及びクラス C の双方からの情報が必要であるとしている．2 本の関連による情報の取得は，AND 条件である．一般的には，クラス B からクラス A 及びクラス C に関連が伸びているからと言って，クラス B が，2 個のクラスから AND 条件で情報を得るとは限らない．後述の様に，AND 条件は，多重度=1 で実現される．尚，情報取得対象クラスが 3 個，4 個の場合にも，AND 条件への要請は同様に存在する．

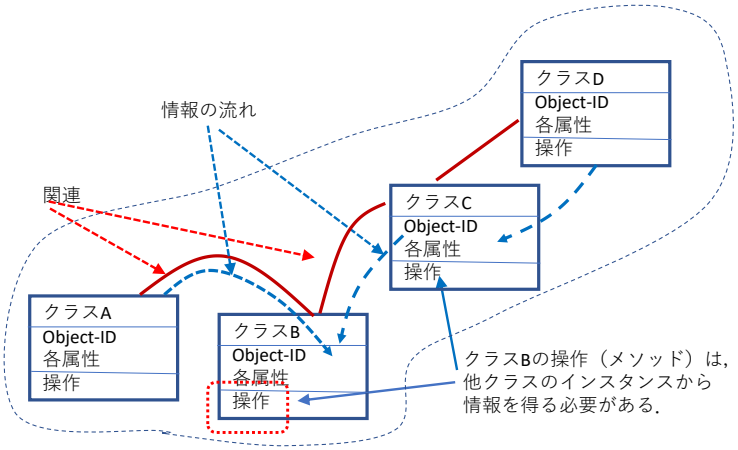


図 4　オブジェクト指向における関連の役割

次に，関連の（クラス A 及びクラス C 側の）多重度について考察する．関連の意味を確認しておく．関連はクラス間に引かれているが，もともと，関連は，2 クラスのインスタンス間のリンクに関する条件である．多重度に関する考察は，インスタンスレベルで行う必要がある．

例えば，図 4 において，（クラス B から見た）クラス A の多重度，及び，（クラス B から見た）クラス C の多重度を，「多重度＝2」と仮定する．2 を選んだのは，「複数」という意味であり，以下の議論は，「多重度＝3」でも成立する．

「多重度＝2」の場合，図 4 のクラス B の 1 つのインスタンスからは，クラス A へ 2 本，クラス C へ 2 本，リンクが伸びる．このリンクを利用して，クラス B の処理プログラムは，クラス A のインスタンス 2 個，及び，クラス C のインスタンス 2 個から情報を取得して，総計 4 個のインスタンスからの情報を用いて，処理を実現することになる．

[2] 関連と言っても，どのような関連かはこの段階では不明である．図 4 では，単なる曲線として関連を表現している．

しかし，クラス毎に，常に 2 個（複数個）のインスタンスを指定する上記の枠組みは，クラス B における業務処理の実現において，使いやすいとは考えられない．例えば，クラス B の処理プログラムは，クラス C の 2 個のインスタンスからの情報をどう順序づけて，計算するのだろうか．そもそも，クラス A なりクラス C から，常に 2 個のインスタンスを取りだす必要があるとは思えない．クラス A やクラス C からは，1 個だけを取り出す方が，自然で使いやすく，煩雑さも無い．

上記の分析から，図 4 の関連の多重度は，クラス A 側及びクラス C 側を「1」とすることが望まれる．「0..1」ではなく，「1..1」であることに大きな意味がある．「１」は，AND 条件を担保する．クラス B のインスタンスが存在する限り，クラス B から，クラス A のインスタンス及びクラス C のインスタンスに，リンクが 1 本ずつ存在せざるを得ない．情報取得に妥当な関連の多重度は，1 対多関連（正確には，少なくとも片方が「1」）が妥当と考える．１対多関連の多用は，ベテランモデラーの行動とも合致している．

尚，図 4 において，クラス C とクラス D の間も 1 対多関連であり，クラス D 側が，「多重度＝1」であるとする．この場合，クラス B から見て，まず，クラス C のインスタンスが決まる．更に，クラス C のインスタンスが決まると，クラス D のインスタンスがユニークに決まる．クラス B から見ると，1 対多関連により「芋づる式」に辿ることのできるクラス D からは，自分の属性値と同様に，属性値を取り出し得る．これも，1 対多関連の特徴のひとつである．本論文では，これを「**１対多関連の連鎖**」と呼ぶ．以下の分析 1 が成立する．

【分析 1】クラス図において，あるクラスに意味的に関係したクラスを特定する際には，1 対多関連（正確には，少なくとも一方の多重度が「1」）の利用が適切である．1 対多なので，リンク先のクラス中の唯 1 個のインスタンスのみが指定される．

ただし，上記の議論で，メソッドを持ち込んだのは，あくまでも，オブジェクト指向における情報の流れを考察するための仮想的なものであり，本論文のクラス図にメソッド（操作）を入れて考察することを意味するものではない．

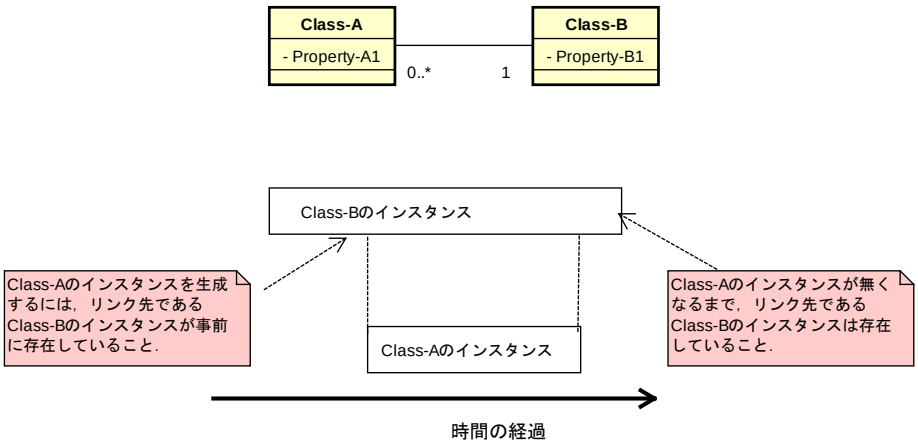


図 5 「1 対多関連」とライフタイム

4.2. 「１対多関連」と時間的前後関係

次に，1 対多関連が，結び付けているクラス間に，時間的前後関係を含意することを論じる．図 5 を用いて説明する．Class-A から Class-B に伸びた 1 対多関連について考える．この場合，Class-A のインスタンスが生まれる前に，対応する Class-B のインスタンスが必ず存在していなければならない．何故ならば, Class-B のインスタンスが存在していないのに, Class-A のインスタンスを作ると,「多重度＝1」ではなく,「多重度＝0..1」になるからである．また，Class-A のインスタンスが存在している間は，リンク先の Class-B のインスタンスを消すことはできない．

尚，1 対多関連には，図 5 において，Class-A のインスタンスを生成した後，リンクされた Class-B のインスタンスを差し替えてもよい「非依存型関連」と，Class-A のインスタンスを生成した後，リンクされた Class-B のインスタンスが固定される「依存型関連」がある．どちらの場合も，本節の議論は成立する．以下の分析 2 が成り立つ．１対多関連は，2 個のクラス間に時間的前後関係という半順序を形成することに注意が必要である．

【分析 2】1 対多関連で接続された 2 個のクラスが与えられた時，「多重度＝1」側クラスのインスタンスは，「多重度＝多」側クラスのインスタンスよりも，時間的前後関係として，事前に存在していなければならない[3]． 1 対多関連は，両端の 2 個のクラス間に時間的前後関係という半順序を形成する．

4.3. 因果関係の存在

前節の分析により，1 対多関連で関係づけられた 2 個のクラスの間には，時間的前後関係がある．そうなると，関連による 2 個のクラスの関係性を，実世界のどのような関係性と解釈するべきなのかが，モデリング上において問題となる．

時間的前後関係の解釈で良く知られた概念に，因果関係がある．因果関係については，古くから，営々たる議論が行われてきたものと想像される．よく知られた条件は，ヒュームによるものであり，以下の 3 点を因果性の 3 要素とした[11].

(1) 2 個の現象が規則的に連接して生起する．
(2) 原因は結果に先行しなければならない（時間的先行性）．
(3) 原因と結果は隣りあった場所になければならない（近接性）．

UML2.5 規格書は，「関係（関連）」について，簡単な定義しか与えていない[4]．しかし，関連が引かれること自体，現象の生起が規則的であり，モデル上で隣りあっているとも見なせる．言うまでもなく，上記 3 要素を満たせば自動的に因果関係が成立するわけではない．しかし，2 個のクラスが関連によってリンクされており，しかも時間的前後関係があるとすると，1 対多関連で表現されるモデルとして，因果関係が成立するとしても，良さそうに見える．以下の分析 3 が成立する．

【分析 3】1 対多関連でリンクされた 2 個のクラスについて，時間的に前のクラスと後のクラスとの間は，原因―結果，材料－製品（中間製品），前提－帰結等の因果関係を含意する．この場合，時間的に前のクラスが原因（材料）であり，時間的に後のクラスが結果（製品・中間製品）である．

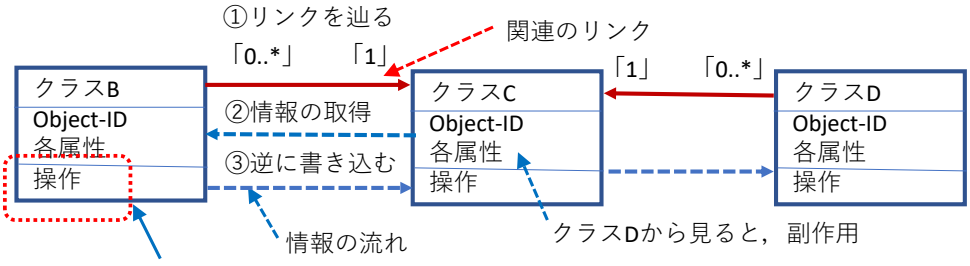


図 6 時間的に前に生成されたインスタンスへの書き戻し

4.4. 副作用の発生

1 対多関連でリンクされた 2 個のクラスについては，時間的前後関係があり，「多重度＝1」側クラスの方が時間的に前である．この場合，時間的に前の「多重度＝1」側クラスのインスタンスの情報を，時間的に後の「多重度＝多」側クラスのインスタンスが利用する場合には，問題はない．しかし，逆に，「多重度＝多」側クラスから，「多重度＝1」側クラスのインスタンスを書き換えると，副作用を生じる恐れがある．

図 6 は，クラス B がクラス C のインスタンスからの情報に基づき，クラス B のインスタンスを生成する際の情報のやり取りを示す図である．まず，クラス B から（時間的に前に生成された）クラス C のインスタンスへ張られた関連のリンクを辿り，クラス C のインスタンスが特定される（図 6 の①）．そして，クラス B は，クラス C のインスタンスから情報を取得してクラス B のインスタンスを生成する（図 6 の②）．

クラス B が，クラス C のインスタンスから情報を読み取るだけ（図 6 の①②）なら，問題は生じない．しかし，クラス B がクラス C のインスタンスを書き換える場合（図 6 の③）は問題である．クラス

[3] 1 対 1 関連の場合には同時となる．この場合でも，後述のハッセ図を構成する際に，半順序に等値を含むのみであり，クラス図のトポロジカルな構造は影響をうけない．
[4] UML2.5 規格書には，以下の様に記述されている．7.8.15 Relationship，Relationship is an abstract concept that specifies some kind of relationship between Elements.

C の当該インスタンスを参照している別のインスタンス（例えば，クラス D のインスタンス）があれば，その動作に副作用を生じさせるからである．以上から，分析 4 が成立する．

【分析 4】1 対多関連でリンクされた 2 個のクラスを考える．「多重度＝多」側のクラスが，「多重度＝1」側のクラスのインスタンスから情報を読み取るだけの場合には，特に問題は生じない．しかし，「多重度＝多」側のクラスが，「多重度＝1」側のクラスのインスタンスに情報を書き込む場合には，時間的に後のインスタンスから時間的に前のインスタンスへの情報の流れとなり，時間的前後関係の方向と情報の流れる方向が逆転する．その結果，時間的に前のインスタンスを参照している他クラスに副作用を与える恐れがある．

4.5. 小括

以上，本章の分析結果を以下にまとめて列挙する．

(1) クラス図の関連としては，1 対多関連の利用が適切である（分析 1）．

(2) 1 対多関連で結ばれた 2 個のクラスの間には，時間的前後関係が成立する．ただし，「多重度＝1」側のクラスが時間的に前であり，「多重度＝多」側のクラスが，時間的に後となる（分析 2）．

(3) 1 対多関連で結ばれた 2 個のクラスの間には，モデル上での因果関係が成立する．「多重度＝1」側クラスが原因（材料）であり，「多重度＝多」側クラスが結果（製品，中間製品）である（分析 3）．

(4) 関連で繋がっている 2 個のクラスにおいて，「多重度＝多」側のクラスが「多重度＝1」側のクラスから情報を読み取り，「多重度＝多」側のクラスのインスタンスを生成するのみであり，「多重度＝多」側のクラスから，「多重度＝1」側のクラスへの書き込みがない場合には問題はない．しかし，「多重度＝1」側のクラスへの「多重度＝多」側のクラスからの書き込みがある場合には，副作用を生じる恐れがある（分析 4）．

(5) 多対多関連は，関連のクラス化により，1 対多関連 2 本に容易に変換できる．1 対多関連の限定利用は，工場モデルクラス図の表現能力を制限するものではない[5]．また，工場モデルクラス図から関係モデルを作成する際にも，1 対多関連を関係（テーブル）上で，容易に表現できる[6]．

5. DAG の推移簡約

本章では，1 対多関連のみを利用する工場モデルクラス図のトポロジー構造が，DAG の推移簡約になることを示す．尚，DAG の持つ特性として，工場モデルクラス図を平面上に描画する際に，まず最上部にクラスを配置し，そこから下方向に向かって，一つ一つクラスを追加する形で，自然に描画できる．この点は，本論文の証明プロセスとは異なるものなので，付録 3 として添付する．

5.1. 有向非巡回グラフ(DAG)

工場モデルクラス図のトポロジー構造を理解し易くするため，グラフ表記を導入する．図 7（左図）はクラス図の例である．関連の多重度は「1」側のみを表示している．一方，図 7（右図）は，そのグラフ表記である．クラスをノード，関連をアークに置き替えた．ただし，アークは矢印であり，「多重度＝1」側を矢印の鏃（やじり）として描いている．前章の分析 2 から，アークは，ノード間の半順序（時間的前後関係）を規定する．アークの矢印の鏃（やじり）側ノードが時間的に前のノード（クラス）となる．

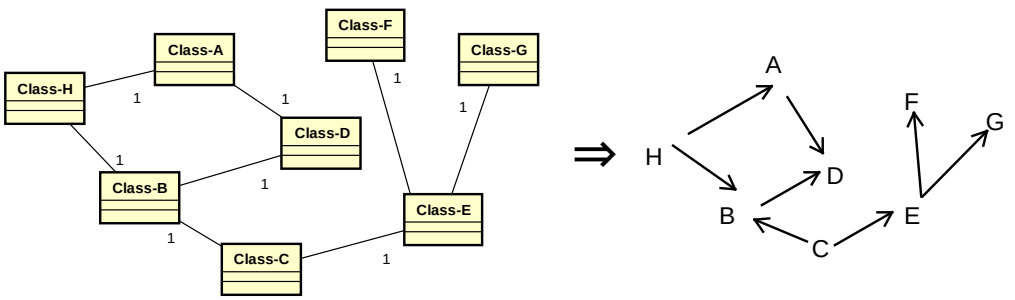


図 7　工場モデルクラス図と抽出されたグラフ構造

[5] 細かい議論なので，説明は付録 1 に示した．

[6] 細かい議論なので，説明は付録 2 に示した．多対多関連を持つクラスは，最終的にデータベース化する際には，付録 1 に示した関連クラスに相当する関係（テーブル）を追加する必要がある．その意味で，多対多関連の排除は，データベース設計上は，デメリットはない．

仮に，ループ状に接続されたアークの矢印の方向がそろって，巡回したとする．これは，不合理である．何故なら，矢印の方向にループを一周すると，どんどん昔に行くとやがて未来に飛ぶこととなり矛盾するからである．よって，工場モデルクラス図は，クラスをノード，関連をアークとする DAG を構成する．以上を分析 5，分析 6 としてまとめておく．

【分析 5】工場モデルクラス図では，1 対多関連の「多重度＝1」方向に辿っても，ループは形成されない．1 対多関連のループが存在する場合はモデリングの誤りと思われる．
【分析 6 】工場モデルクラス図は，クラスをノード，関連をアークと見なして，グラフ構造化すると，DAG となる．

尚，クラス図におけるクラスをノード，関連をアークと見なすことにより，クラス図をグラフとして認識できる事は，工場モデルクラス図に限定された性質ではない．しかし，工場モデルクラス図では，アークを 1 対多関連に限定して，時間的前後関係と言う，半順序関係をアークに持ち込んだ．その結果として，トポロジカルソートにより，時間的前後関係という理解し易い尺度を用いて，ノード（クラス）を順序付け可能としている．

5.2. 「1 対多関連」の推移簡約化

1 対多関連では，「多重度＝多」側クラスのインスタンスを決めると，「多重度＝1」側クラスのインスタンスがユニークに決定される．これは，関数従属に類似した概念である．従って，工場モデルクラス図では，推移的関数従属と同様の問題が生じる．推移的な関数従属に該当する 1 対多関連は，冗長であり，工場モデルクラス図から排除する必要がある．

図 8 (左図）は，3 個のクラス（クラス A，B，C）において，2 個の 1 対多関連が，冗長な 1 対多関連を伴う例である．簡明化のため，グラフ表記した．クラス A(ノード A)のインスタンスが決まれば，クラス B（ノード B）のインスタンスが決まる．そして，クラス B（ノード B）のインスタンスが決まればクラス C（ノード C）のインスタンスが決まる．結果的に，クラス A(ノード A)のインスタンスが決まれば，クラス C（ノード C）のインスタンスが決まる．この関係を A→B→C[7]と表記する．図 8 (左図）の A→C は，推移的で冗長である．モデリングに際しては，図 8 (右図)の様に，冗長を除去するべきである．冗長アークの除去は，ノード間の半順序が構成する DAG の推移簡約化に等しい．

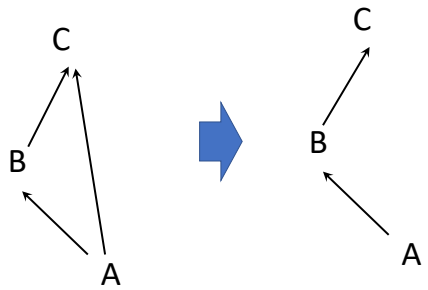


図 8　推移的な 1 対多関連と冗長表現

ここで，「推移簡約化」とは，（図 8 左図にあるような）冗長アークを除去することを言う．結果として, 図 8 右図に示すような, 冗長アークを持たない DAG へ変換される. これを実現するためには, A→B, B→C が，確実に，モデルとして取り出されていなければならない．A→B，B→C を構成する 2 個の 1 対多関連（アーク）は，それぞれ，これ以上，細かい 1 対多関連の組み合わせには分解できないものとする．これは非推移的な 1 対多関連に該当する．以下の分析 7 が成立する．

【分析 7】1 対多関連を用いて，対象ドメインをモデル化する際には，対象ドメインにおいて，それ以上，関係性を細分化できないような，直接的な関係を持つ「関連」とする必要がある．そして，グラフ構造上では，冗長なアークを削除することが望ましい．

例外的ではあるが，図 8（左図）のような，冗長に見えるグラフ構造が必要とされる場合もある．ク

[7] →は，関数従属性を示す．A→B の意味は，「クラス A のインスタンスを選ぶと，クラス B のインスタンスがユニークに決まる」である．

ラス A の特定のインスタンスから出発して，A→B→C で指定されるクラス C のインスタンスと，A→C で指定されるクラス C のインスタンスが異なる場合である．

一方，図 9 の様に，2 個のルートが再収斂した場合については冗長ではない．図 9 の右ルートはクラス D(ノード D)に関する処理を持っているが，左ルートは，それを代替できない．両ルートはそれぞれ独立した意味がある．ただし，A→B→C で指定されるクラス C のインスタンスと，A→D→C で指定されるクラス C のインスタンスが異なるのか同一なのかは意識する必要がある．

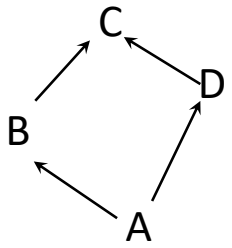


図 9　再収斂の存在する場合

5.3. ハッセ図としての工場モデルクラス図

本節では，4.1 節から 5.2 節までの分析を総合し，関連として 1 対多関連のみを利用するクラス図（工場モデルクラス図）のトポロジー構造が，DAG の推移簡約に等しいことを確認する．ただし，工場モデルクラス図のグラフ表記は，クラスをノードに，1 対多関連をアーク（矢印）に置き替え，「多重度＝1」側を矢印の鏃（やじり）で表現したものである．

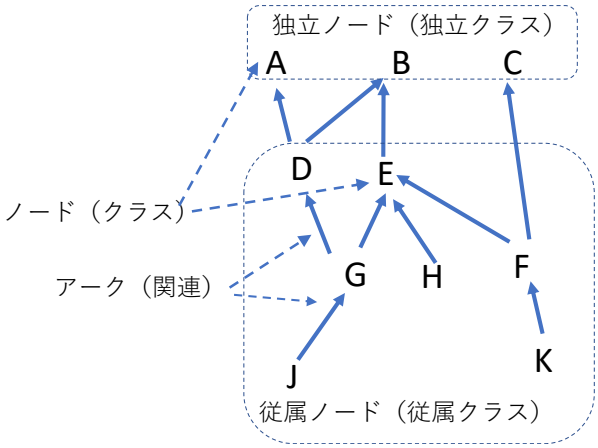


図 10　工場モデルクラス図のグラフ表記

1 対多関連のみを関連として用いたクラス図が，DAG の推移簡約となることの証明概要は，3.3 節に示した．まとめると以下の様になる．また，分析 8 が成立する．

【DAG の推移簡約であることの証明】

関連として 1 対多関連のみを利用するクラス図を考える．クラス図の各クラスをノード，1 対多関連をアークとしてグラフ構造を考える．この時，分析 1，分析 2 から，アークは両端のノード間の時間的前後関係（半順序）を構成する．従って，グラフは，有向グラフである．アークの向きは，多重度=1 の方向を矢印の鏃（やじり）で表現する．加えて，モデリングが正しければ，グラフは巡回を持たない（分析 5，分析 6）．グラフは，有向非巡回グラフ（DAG, Directed Acyclic Graph）となる．しかも，冗長アークを取り除くことにより DAG は，推移簡約となる（分析 7）．以上から，関連として 1 対多関連のみを用いる工場モデルクラス図は，DAG の推移簡約となる．

【分析 8】工場モデルクラス図は，クラスをノード，関連をアークとするグラフ表現すると，DAG の推移簡約を構成する．ただし，アークは「多重度＝1」側を鏃（やじり）とする矢印で表現する．

図 10 に工場モデルクラス図のグラフ表記イメージを示す．DAG の推移簡約を描画したものは「ハッセ図」と呼ばれる[12]．DAG では，アークが到着するのみで，出て行かない独立ノードが少なくとも 1 個存在することが知られている．描画に際しては，最上部に「独立ノード」を並べる．独立ノード以外

の「従属ノード」は，その下に次々とぶら下げることになる．このグラフ構造は，そのまま工場モデルクラス図に継承される．工場モデルクラス図のサンプルや用語説明については，3.1 節を参照されたい．

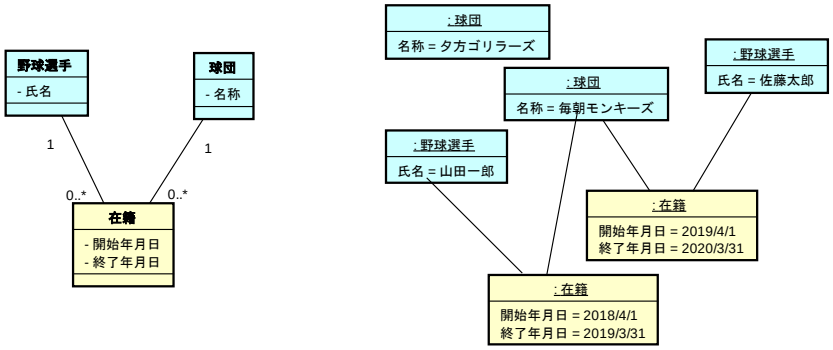


図 11　工場モデルクラス図の簡単な例（I）

5.4. 工場モデルクラス図の具体例

図 11 には，工場モデルクラス図の極めて簡単な例とそのオブジェクト図を示した．図 11（左図）においては，「1」側の 2 個のクラス「野球選手」「球団」が上流側クラス（材料）となり，下流側クラス（製品）である「在籍」クラスが生成されている．野球選手が特定され，かつ，所属球団名が確定しない限り，「在籍」インスタンスは生成できない．図 11（右図）は，オブジェクト図で見た，実現例である．

この小さな例でも，クラス図は，因果関係を持っている．野球選手と球団がそれぞれ存在していることが前提条件である．これが存在しないと，在籍と言う，新たな結果を記録することはできない．野球選手が誰であるかが未定では，在籍の記録など作れない．

オブジェクト図レベルで見ると，「在籍」のインスタンス 1 個から，2 本のリンクが伸びて，それぞれ，「野球選手」インスタンスと，「球団」インスタンスにリンクされている．ひとつのインスタンスから 2 本のリンクが伸びているので，AND 条件となる．つまり，「野球選手」と「球団」のインスタンスは，両方とも存在していないと，リンクは引けない．

尚，上記の「在籍」クラスを表現するのに，関連クラス[8]を用いることも考えられる．しかし，実装時まで考慮すると，関連クラスは「関連のクラス化」と等価と思われる．このため，本論文では，関連クラスの議論には立ち入らない．

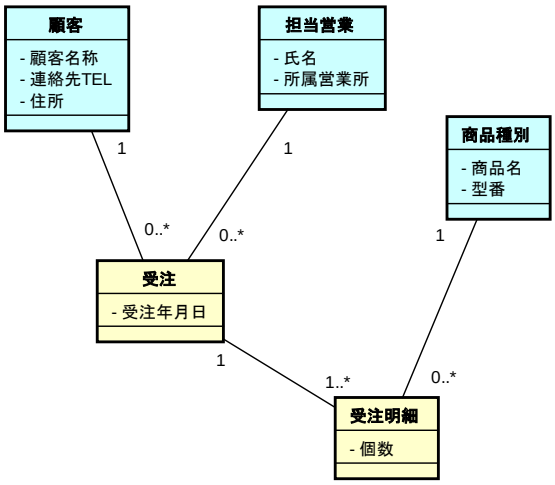


図 12　工場モデルクラス図の簡単な例（II）

図 12 は，工場モデルクラス図の別の簡単な例である．対象ビジネスドメインから「顧客」「担当営業」「商品種別」「受注」「受注明細」の 5 個のクラスを取り出している．他のクラスのデータ状態に無関係

[8] UML2.5 の関連クラスの規定では，関連クラスが複数のインスタンスを持つことを，「Note」として，認めている．しかし，これでは，関連のクラス化と，関連クラスとの間に，働きとしての差異はない．関連クラスという特別のシンタクスを導入した意味が無いのではないかと思われる．その意味でも，本論文では，関連クラスには立ち入らない．

に，インスタンスを追加できるのは，「顧客」「担当営業」「商品種別」クラスである．そこで，この 3 個のクラス（独立クラス）を最上部に並べた．次に，受注したとする．この段階では，顧客からの受注が明らかになっただけで，商品種別は確定していない．そこで，まず，「顧客」クラスからの情報と，「担当営業」クラスからの情報に基づいて「受注」クラスを作成する．次に，この「受注」クラスの存在を前提にして，「商品種別」クラスからの情報に基づき，「受注明細」クラスを追加する．この様に，時間的進行につれて，次々とクラス（従属クラス）を作成してゆく．

6. 汎化関係の導入

6.1. 汎化関係の導入

本章では，工場モデルクラス図に汎化関係を導入する．工場モデルクラス図は，関連（集約・コンポジションを含む）とクラスから構成されている．関連・集約・コンポジションは，2 クラス間に直線を引いて表現するが，規定しているのは，2 個のクラスにそれぞれ属するインスタンスの間に張られるリンクに関する制約である．このため，関連・集約・コンポジションは多重度を持つ．一方，汎化関係は，同じ様に，スーパークラスとサブクラスとの間に矢印を引いて表現されるが，クラス間の関係性を規定するだけであり，インスタンス間のリンクに関する規定ではない．このため，多重度は持ち得ない．汎化関係と関連は，全く性質が異なる．

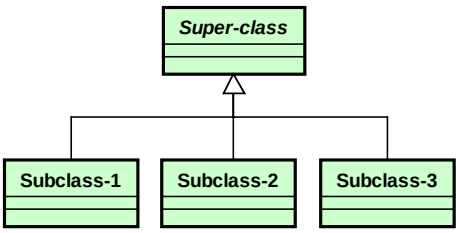


図 13　汎化関係を持つクラス

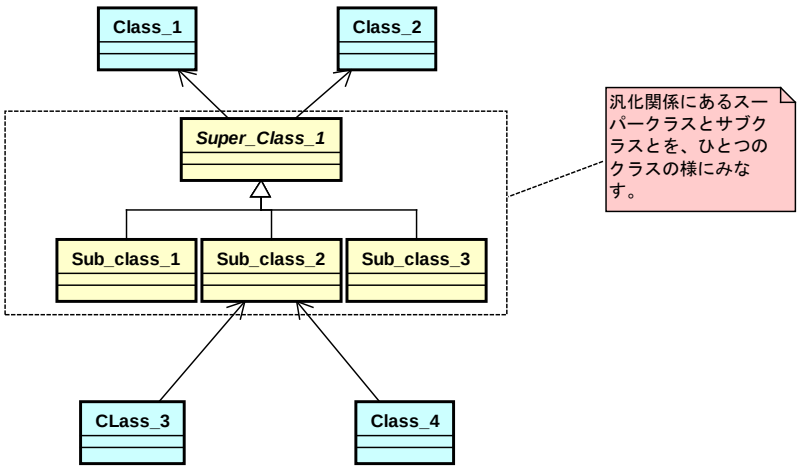


図 14　汎化関係全体のクラス化

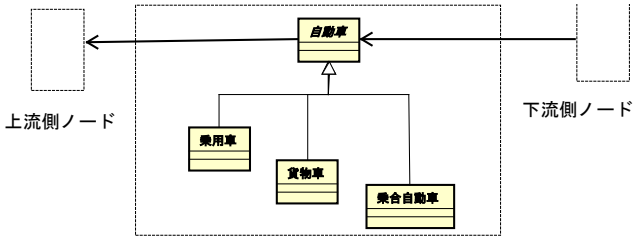
図 13 は，汎化関係の例である．スーパークラスは抽象クラスとしている．サブクラスはインスタンスを持つ可能性がある．サブクラスのインスタンスは，スーパークラスから見ても存在する．しかし，スーパークラス中にインスタンスがコピーされる訳では無い．サブクラスのインスタンスとスーパークラスのインスタンスの間にリンクが引かれることもない．

スーパークラスとサブクラスとの間の時間的前後関係について考えてみたい．サブクラスに追記されたインスタンスは，同時にスーパークラスにおいても見える様になる．一方，サブクラスのインスタンスを消去すると，同時に，スーパークラスからも消える．明らかに，スーパークラスとサブクラスとの間には，時間的前後関係は存在しない[9]．更に，スーパークラスとサブクラスは，主キーとして同一のも

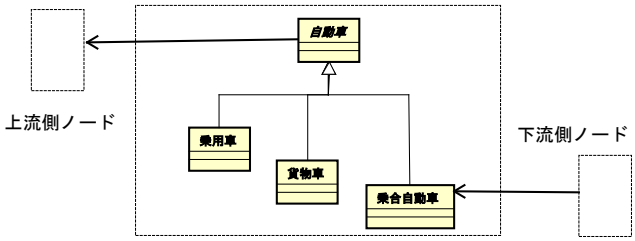
[9] 広く知られた Composite パタンにおいて，汎化関係に「集約」が付加されているのも，偶然とは思えない．集約は，親（全体）と子（部分）の間に同時性を要求するからである．

のを利用する．従って，スーパークラスとクラスとの間に時間的前後関係を考えるのは無理がある．
以上の分析から，工場モデルクラス図のクラスの位置に，汎化関係のスーパークラスとサブクラスを持つ汎化関係をまとめて導入することが適切と判断する．クラス間の 1 対多関連は，0 又は 1 本に限定するのは，工場モデルクラス図の基本的な方針である．従って，図 14 の様に，汎化関係がクラスの位置に入っても，この規定は変化することはない．次の分析 9 が成り立つ．

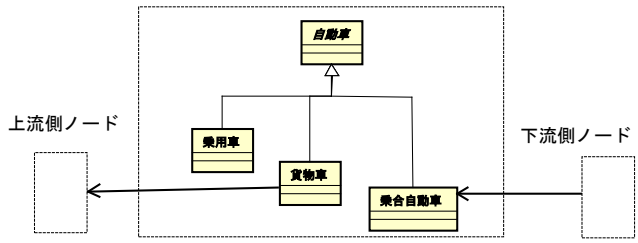
【分析 9】工場モデルクラス図では，設計思想として，クラス間に高々 1 本の 1 対多関連を引くことしか許していない．クラスの位置に汎化関係を入れた場合でも，ノード（クラス）とノード（クラス）の間に存在し得る 1 対多関連の本数は，高々，1 本である．



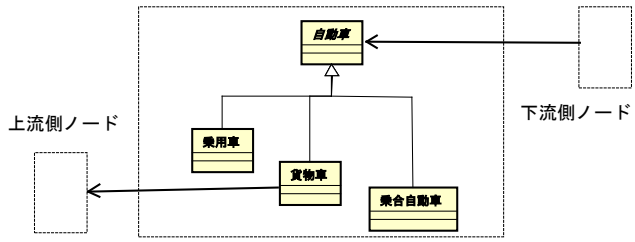
（1）上流側ノードへはスーパークラスから出発、下流側ノードからもスーパークラスに到着



（2）上流側ノードへはスーパークラスから出発、下流側ノードからはサブクラスに到着



（3）上流側ノードへはサブクラスから出発、下流側ノードからもサブクラスに到着



（4）上流側ノードへはサブクラスから出発、下流側ノードからはスーパークラスに到着

図 15　スーパークラス／サブクラスの利用区分

6.2. 「１対多関連」の連鎖

分析９の条件の下でも，１対多関連の連鎖が実現されることを確認する．ただし，工場モデルクラス図のクラスの位置を，「ノード」と呼ぶことにする．ノードには，クラス或いは，汎化関係が入る．汎化関係が入っているノードに対しては，下流側からの関連が，スーパークラス又はサブクラスに到着する．更に，当該ノードから上流側に伸びる関連も，スーパークラス又はサブクラスから出発する．従って，汎化関係を持つノードにおける，１対多関連の連鎖パタンは，４通りある．それぞれについて，１対多

関連の連鎖が途切れないことを確認する．図 15 は，その 4 通りのイメージを示す．例として自動車を用いているが，事例のドメイン選択に深い意味はない．

図 15（1）を確認する．1 対多関連は，スーパークラスから上流側ノードへ向かう．上流側ノードから情報を得て，業務処理プロセスにおいて生成されたインスタンスは，何れかのサブクラスに保存される．このサブクラス中のインスタンス情報は，スーパークラスに 1 対多関連を伸ばしている下流側ノードから読み取られる． 1 対多関連の連鎖が成立している．

次に，図 15（2）の場合は，下流側ノードからの 1 対多関連はサブクラスに到達する．上流側ノードへは，スーパークラスから出て行く．スーパークラスからは，どのサブクラスのインスタンスも見えているので， 1 対多関連の連鎖は成立している．

図 15（3）の場合には，下流側ノードから 1 対多関連がサブクラスに到達し，上流側ノードへも，当該サブクラスから出て行く．もし，到達するサブクラスと出発するサブクラスが異なるクラスであれば，連鎖が途切れる．意味的に当然と考える．一方，同一のサブクラスのインスタンスを経由した 1 対多関連の連鎖には何の問題もない．

最後に，図 15（4）について考える．下流側ノードから 1 対多関連がスーパークラスに到達し，上流側ノードへは，サブクラスから出て行くケースである．まず，下流側ノードからの関連の到着であるが，スーパークラスに到着している．しかし，スーパークラスには，インスタンスは置けないので，インスタンスは，どこかのサブクラスに存在するインスタンスにリンクが到着しているはずである．上流側ノードへの連鎖は，当然，このインスタンスからしか出て行けない．インスタンスを経由した 1 対多関連の連鎖は問題ない．

以上の分析から，ノード間の関連の本数を 1 としても， 1 対多関連の連鎖は途切れないことを確認できた．ただし，実際の問題を，図 15 のどの場合を選択して表現するかは，具体的なドメインの業務を想定した上で，選択する余地があるかもしれない．

7. クラス間の関係を表現できない場合[13][14]

本章では，工場モデルクラス図では表現できない場合の対処方法について議論する．ただし，表現できない場合の対処方法は，技術者の技術的選択の範囲内と思われる．本章は，一般性を持つオリジナリティを主張するものではない．工場モデルクラス図の参考と考えて頂きたい．

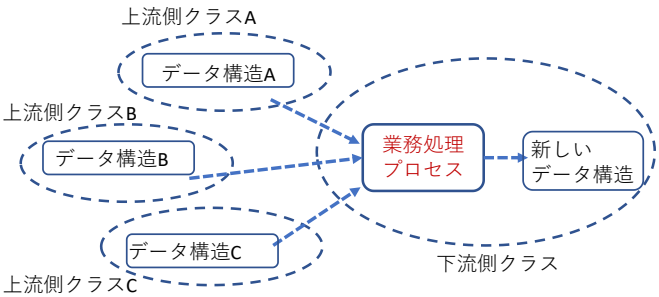


図 16 表現可能なクラス間の関係性

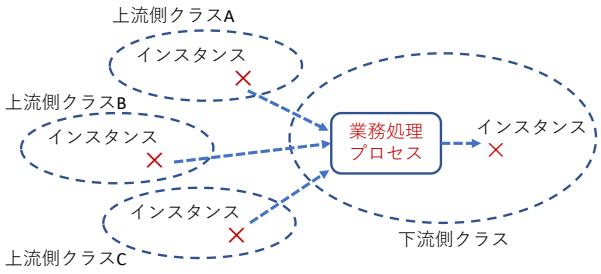


図 17 処理イメージで描いたクラス間の関係性

7.1. クラス間の関係を表現できる場合

本題に入る前に，「工場モデルクラス図により，クラス間の関係を表現できる場合」を確認する．図 16 には，表現できる場合の情報の流れを示した．上流側クラスから得られる情報に基づいて，下流側クラスでは業務処理プロセスの実行が想定され，結果を保存するためのデータ構造が，下流側クラスとして設計される．データ構造は，クラスの属性，プライマリーキー（あるいは，オブジェクト ID）等を含ん

でいる．

図 16 に対して 1 対多関連を適用し，実際のアプリケーションプログラムにおける処理イメージで示すと，図 17 の様になる．上流側クラスからインスタンスを取り出せるのは，クラス毎に 1 個である（図 17）．そのインスタンスが持つ情報に基づいて，業務処理プロセスが実行され，結果として，下流側クラスのインスタンスが新たに作成される．

7.2. クラス間の関係を表現できない場合

本題に入る．工場モデルクラス図では表現できない場合を考える．表現できる場合である「上流側クラスから利用できるインスタンスは，各クラス 1 個のみであり，かつ，情報の流れは，上流側から下流側へと 1 方向である．」が満たされない場合である．即ち，(I) 同一の上流側クラスから取り出されるインスタンス数が 2 個以上となる場合，又は，(II) 情報の流れが双方向となる場合，である[10]．上記 (I) は分析 1 の否定，上記 (II) は，分析 4 に相当する．

最初に上記 (I) について考える．同一上流側クラスから複数のインスタンスを取り出す場合が問題となる．処理イメージを図 18 に示す．例えば上流側クラスから 1 週間分のインスタンスを取り出して，合計値を求めて，新しいインスタンスとして，下流側クラスに週毎に登録する場合である．このようなクラス間における関係は，1 対多関連のみでは表現できない．このような場合を本論文では「**統計型**」と呼ぶ[11]．

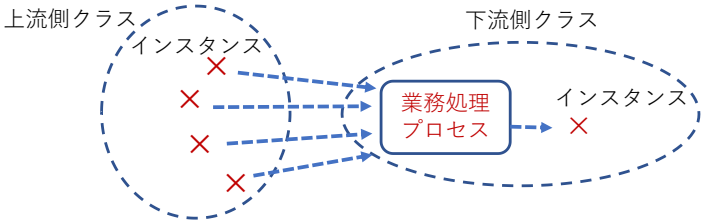


図 18　表現できない統計型の処理イメージ

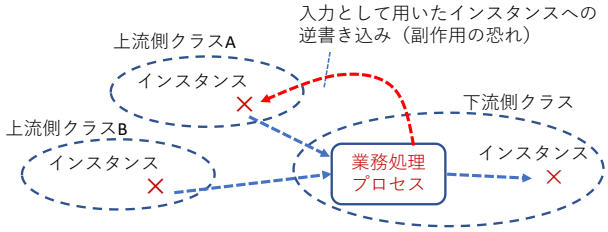


図 19　表現できない在庫型の処理イメージ

次に，上記 (II) について考える．分析 4 によれば，図 19 の様に，下流側クラスが，上流側クラスのインスタンスを更新する場合は，表現できない．図 19 では，情報の流れが，上流側クラスのインスタンスから，下流側クラスに流れ，更にその後，下流側クラスから上流側クラスのインスタンスに逆戻りしている．つまり，クラス間で，情報が行ったり来たりしている．このような現象は，1 対多関連のみでは表現できない．代表的な例は，在庫数の様に，時間の経過とともに，過去に作成されたインスタンスが何度も更新される場合である．このような場合を本論文では「**在庫型**」と呼ぶ．

7.3. クラス間の関係を表現できない場合の具体例

本節では，クラス間の関係を十分には表現できない，統計型と在庫型の具体例を示す．

【統計型】

図 20 に，クラス間の関係が，統計型である場合の例を示す．営業所毎に，各営業日の売上データを保持する「売上」クラスがあり，その「売上」クラスのインスタンス群から，各営業所の週毎の売上総額を計算して，「売上合計」クラスとして記録したい．

本来なら「売上合計」クラスから，「売上」クラスへ向けて，関連を引きたい．しかし，営業所毎に，週単位の売上合計を計算するには，「売上」クラス中のインスタンスの中から，1 週間分のインスタンスを特定する必要がある．これは，1 対多関連では表現できない．

10　下流側クラスから上流側クラスに情報が流れる場合も考えられるが，この場合は，入力と出力のクラスを交換すれば，「情報の流れは，上流側クラスから下流側クラスへと 1 方向である．」となるので除外した．

11　統計型の処理は，一種の VIEW なので，概念モデリングには持ち込むべきではないとの考え方もあると思われる．妥当な考え方と思われるが，本論文では，説明の都合上，エンティティ層で扱うこととする．

しかし，このままでは，「売上合計」クラスと「売上」クラスの関係性が見えない．そこで，図 20 の場合には，「売上合計」クラスから「営業所」クラスへ 1 対多関連を引いた．「売上合計」クラスの上流側クラスであるところの，「売上」クラスの，更に上流側のクラスである「営業所」クラスに対して，1 対多関連を引いている．この様に，あるクラスの上流側クラスに対して，クラス図の関連を引けない場合，更にそのまた上流のクラスへ関連を張ることは，有効な手法かもしれない．「売上合計」クラス中のインスタンスを検索する場合の手がかりになる．しかし，クラス間の関係を表示する方法としては不完全である．

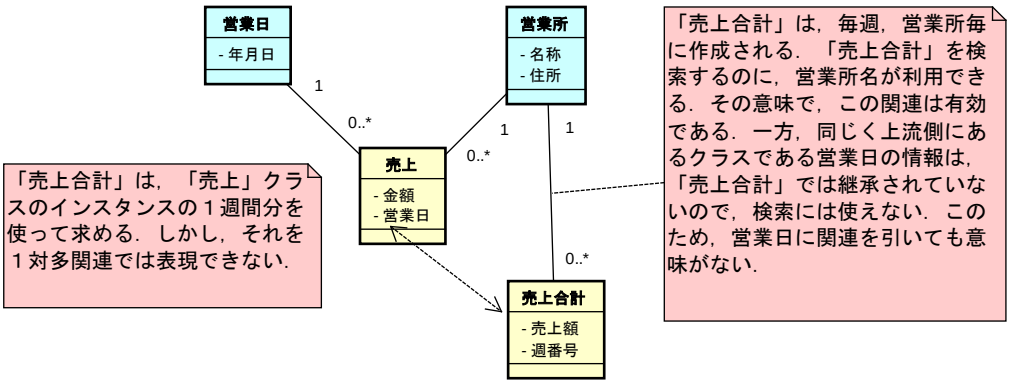


図 20「統計型」クラスの例

尚，ここで，図 20 のクラスの属性について触れておく．例えば，図 20 の売上合計には，「営業所」を示す属性はない．しかし，「売上合計」インスタンスがどの営業所のインスタンスかは，1 対多関連のリンクから一位に判明する．この様に，1 対多関連の連鎖により，上流側クラスに属するリンク先インスタンスが確定するので，「売上合計」クラスに「営業所」属性を設ける必要は無い．

【在庫型】

次に，クラス間の関係が，在庫型である場合の例を図 21 に示す．図 12 のクラス図が既に完成していたとして，このクラス図に，クラスを追加する．図 12 のクラス図に対して，まず，「商品種別」クラスの下流側クラスとして，「在庫」クラスを追加する．「在庫」クラスは，「商品種別」毎に「受領済」「引当済」「出庫済」を属性として保持する．この段階では，問題は生じない．図 21 では，「受領済」とは，当該商品の今までの受領数の合計という単純なイメージである．「引当済」「出庫済」も同様に，単なる累計である．

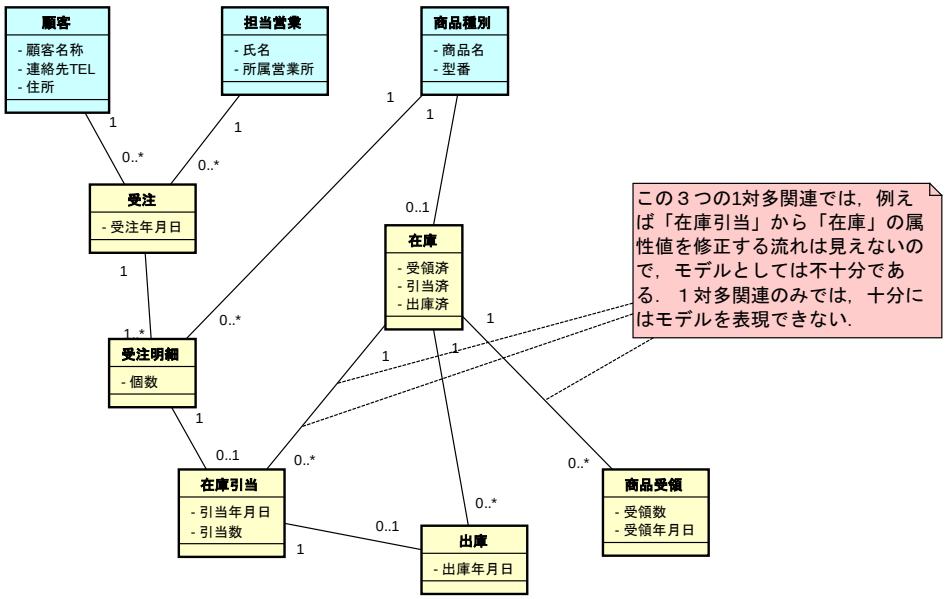


図 21「在庫型」クラスの例

次に，「在庫」クラスを上流側クラスとするクラスが作成される．具体的には，メーカーから商品を受領したことを，日々記録する「商品受領」クラスを，「在庫」クラスの下流側クラスとして作成する．「商

品受領」クラスは，商品毎に，メーカーからの受取りを記録し，「在庫」クラスをアップデートする．「商品受領」クラスと「在庫数」クラスとの間の関係は，毎日変化する商品受領状況をリアルタイムに反映する在庫型である．

「在庫」クラスの「受領済」属性をアップデートするためには，「商品受領」クラスが，上記「在庫」クラスから，現状における受領済（数）を取得する必要がある．この段階での情報の流れに問題はない．上流側クラスである「在庫」クラスから，下流側クラスである「商品受領」クラスへと情報が流れるからである．尚，「在庫」の下流に，商品の受領を表す「商品受領」を作成した．これは，在庫数＝0 で，在庫インスタンスがスタートして，そこに後から商品受領が始まると考えたからである．

しかし，「商品受領」クラスは，これだけでは業務処理プロセスを終了できない．受信した「受領済」属性値に商品の受領数を加算し，「在庫」クラスへ書き戻す必要があるからである．即ち，下流側クラスから上流側クラスへの，逆方向の情報の流れが発生する．このようなクラス間の関係は，1 対多関連のみでは，表現できない．この問題を解決するひとつの方法は，図 21 でも用いている様に，テキストを書き込む「ノート」，及び，図要素とノートをつなぐ破線「アンカー」を利用することであろう．「ノート」に副作用に注意することを明記することになる．

尚，この後に作成される「在庫引当」クラスと「在庫」クラスとの間の関係，更にその後に作成される「出庫」クラスと「在庫」クラスとの間の関係についても，同様に在庫型の関係が成立している．

8. まとめ

本論文では，著者らが提案している 1 対多関連のみを利用するクラス図（「工場モデルクラス図」と呼称）に関する証明を大幅に見直した．具体的には，オブジェクト指向の原点に立ち帰り，「クラス図に必要な関連とは何か」からスタートした．証明は直線的であり，短くはないが，大幅に簡明化できたと考える．この簡明性によって，工場モデルクラス図の背後にある設計思想を理解して，クラス図を作成することが可能になると考えたい．

加えて，本論文では，従来は除外されていた，汎化関係を工場モデルクラス図へ導入した．工場モデルクラス図が対象としてきた，関連・集約・コンポジションは，2 個のクラスのインスタンス間の関係を規定するものであり，多重度を持つ．これに対して，汎化は，クラス間の関係を規定しており，多重度とは無縁である．従って，工場モデルクラス図のアーキテクチャを壊さない様に，汎化は，工場モデルクラス図のクラスの位置に置くべきと考えた．

本論文のひとつの目的は，「クラス図を如何にして作成するのか」を初学者のソフトウェア・エンジニアでも十分に理解できる方法論として提示することである．その意味では，工場モデルと言う，一種のテンプレートの提示と表現可能範囲の明確化には意義があると考える．

尚，椿正明は，エンティティ（クラス）を，「リソース」「イベント」「要約」「在庫型」「断面」に区分している（文献[10]，p.32）[12]．「リソース」は本論文の独立クラスに等しく，「イベント」は従属クラスに等しいと思われる．また，「要約」は本論文の「統計型」，「在庫型」は本論文の「在庫型」に近い．「断面」は，「在庫型」の結果を「統計型」で処理しているとの印象がある．本論文は，椿による経験的方法論の正当性を傍証していると考えている．

謝辞

本論文は，井田明男博士とのクラス図についての共同研究の成果を，著者が見直して整理したものである．井田明男博士に謝意を表したい．また，勝野裕文博士には，本論文のドラフトに対して，多くの重要なコメントを頂いた．深く感謝したい．ただし，本論文中の誤りの責任はすべて著者にある．

参考文献

[1] 金田重郎, 井田明男, “ハッセ図構造を持つ実体関連モデルと正規化データベース,” 情報システム学会論文誌, Vol.16, No.1, pp.17-29, 2020 年 9 月.

[2] 金田重郎, 井田明男, “ハッセ図構造を持つ実体関連モデルと第 3 正規形データベース,” 第 15 回情報システム学会全国大会・研究発表大会, 2019 年 11 月.

[3] 金田重郎, 井田明男, “ハッセ図構造を持つ実体関連モデルと第 3 正規形データベース,” 情報処理学会, 情報システムと社会環境研究会, 2019 年 11 月.

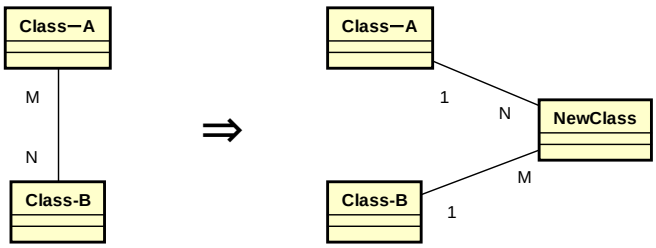
[4] 金田重郎, 井田明男, 森本悠介, “ハッセ図としてのクラス・ER 図：クラス図作成のためのガイドラ

[12] 椿の分類の概略は以下の通りである．「リソース」はビジネスに必要となる「もの」．「イベント」は出来事の記録，「要約」は「イベント」の数値属性を対象とした平均・合計等，「在庫型」はイベントにより常時変化する数値属性，「断面」はリソースや在庫型のある時点を保存するエンティティ．

イン,” 電子情報通信学会・知能ソフトウェア研究会, 2019 年 3 月.
[5] 金田重郎, 井田明男, 森本悠介, “ハッセ図としてのクラス図・ER 図について,” 第 14 回情報システム学会全国大会・研究発表大会, 2018 年 12 月.
[6] 金田重郎, 井田明男, 森本悠介,“正規化クラス図（ER 図）作成のガイドラインー要のもの・こと間のハッセ図としてのクラス図ー,” SES2018 シンポジウム論文集, pp.93-102, 情報処理学会, 2018 年 9 月.
[7] 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ンー行為文と関数従属性に基づくクラス図の作成ー,” 電子情報通信学会論文誌 D, Vol.J98-D, No.7, pp.1068-1082, 2015 年 7 月, DOI: 10.14923/transinfj.2014JDP7103.
[8] 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 年 7 月.
[9] 平澤章, “UML モデリングレッスン,” 日経 BP, 2008 年 1 月.
[10] 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 年 11 月.
[11] スティーヴン・マンフォード, ラニ・リル・アンユム,“哲学がわかる 因果性,” 岩波書店, 2017 年 12 月.
[12] 小野寛晰, “情報数学講座（全 15 巻）第 2 巻　情報代数,” 共立出版, 1994 年 2 月.
[13] 金田重郎, “ハッセ図（工場モデル）に基づくクラス図作成手法の適用性,” 第 16 回情報システム学会全国大会・研究発表大会, 2020 年 12 月.
[14] 金田重郎,“ハッセ図（工場モデル）に基づくクラス図作成手法の適用性,” 電子情報通信学会・知能ソフトウェア工学研究会, 2020 年 9 月.
[15] 増永良文, “リレーショナルデータベース 入門―データモデル・SQL・管理システム・NoSQL,” サイエンス社, 2017 年 3 月.
[16] データ総研, “エンティティの分類,” http://www.drinet.co.jp/blog/kurosawa/2006/エンティティの分類.html (ただし, 椿正明の分類[10]に対して, 1 種類追加されている.) 2022 年 5 月 3 日参照.

付録１　関連のクラス化

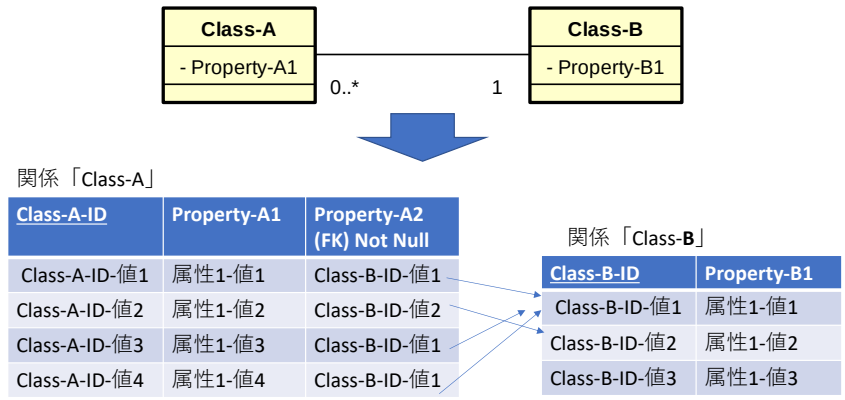
本論文のオブジェクト指向モデリングでは，関連として 1 対多関連のみ利用する．しかし，この条件は，工場モデルクラス図の表現能力を制約しない．何故ならば，任意の多重度を持つ関連は，付図 1-1 に示す様に，新しいクラス（付図 1-1 では NewClass）を導入すると，1 個の新しいクラスと，2 個の 1 対多関連に変換できるからである．本論文では，この追加したクラスを「関連のクラス」と呼ぶ．尚，本付録は，工場モデルクラス図において，1 対多関連のみを利用すればどのようなクラス図でも描けることを主張しているが，それは「1 対多関連のみをクラス図において利用する」ことの第一義的な理由ではない．



付図 1-1　任意の多重度を持つ関連の「関連のクラス」による置換

付録２　関係モデル上の表現

作成されたクラス図は，通常，関係モデルに変換してデータベース化されると思われる．この場合，クラスと関係（テーブル）を 1 対 1 に対応させるのが常識的なアプローチである．付図 2-1 には，1 対多関連を関係モデルに変換する一般的手法を例示する. 付図 2-1 の上部のクラス A 及びクラス B の間の 1 対多関連は，クラス A を変換した関係（テーブル）の中で，FK（Foreign Key）として表現される．ここで，FK が格納される属性が，「Not Null」でなければならない．仮に，Null を許すと，Class-A 側のタプルが 1 個指定されても，Class-B 側のタプルが指定されない．その結果，1 対多関連の「1」が「0..1」に変化するからである．

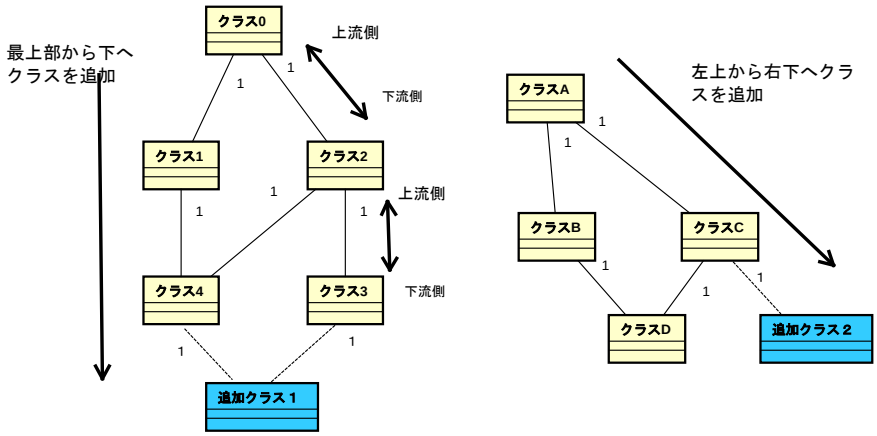


付図 2-1　FK（外部キー）を用いた 1 対多関連の実装

付録３　2 次元平面への描画アルゴリズム

工場モデルクラス図のトポロジー構造は，DAG の推移簡約であるため，二次元上にクラス図を描画する際には，描画・修正が容易であるとの特長がある．本付録３では，描画が容易である理由を概略説明する．

工場モデルクラス図を 2 次元平面上に描画する際には，付図 3-1 の左図に示す様に，1) 最初に作成したクラスを，2 次元平面の最上部に配置し，以後，2) 下方向へ向かって，逐次，クラスを追加してゆけば描画できる．クラスの追記に際して，既配置の各クラスの位置に，大きな修正は不要である．付図 3-1 の右図に示す様に，左上から右下に流れる形で配置しても，描画の容易さは同様である．クラス図を読み取る際にも，付図 3-1 の配置は都合が良い．



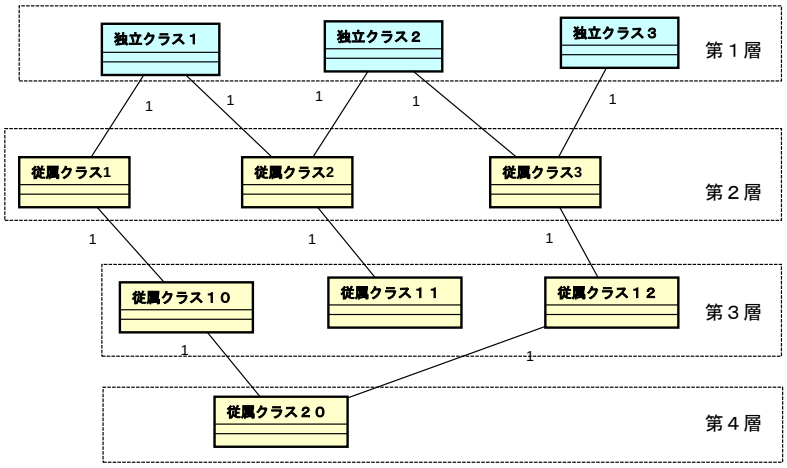
付図 3-1　工場モデルクラス図の 2 次元平面上への描画

工場モデルクラス図のトポロジー構造は，DAG の推移簡約である．DAG のノードは，トポロジカルソート可能であることが知られている．「トポロジカルソート」とは，工場モデルクラス図で言えば，クラスを順序付けして，どのクラスもそのクラスと関連で結ばれた下流側クラスより前に来る様に並べることを言う．トポロジカルソートによって，クラスは，上流側から下流側へ向かって，順序付けられる．ただし，トポロジカルソートしただけでは，クラスは一列に並んでいるだけであり，これを 2 次元平面上に配置する必要がある．尚，トポロジカルソートのアルゴリズムは， Wikipedia 等でも解説されているので，本付録では立ち入らない．

次に，トポロジカルソートされたクラスを，2 次元平面上に配置する．付図 3-2 は，工場モデルクラス図の例である．付図 3-2 の全クラスに対するトポロジカルソートの処理結果は，以下のようなものとなる．

独立クラス 1，独立クラス 2，独立クラス 3，従属クラス 1，従属クラス 2，従属クラス 3，
従属クラス 10，従属クラス 11，従属クラス 12，従属クラス 20

トポロジカルソートのみでは，クラスの順序が決まっただけなので，2 次元平面上にクラス図を描く時のクラスの配置に読み替える必要がある．配置アルゴリズムも種々考えられると思われるが，本付録 3 では，簡明な手法として，以下のアルゴリズムを適用する．



付図 3-2　工場モデルクラス図の例(層を仮定)

【クラス配置のアルゴリズム(概要)】

2 次元平面上のクラスの位置に関して，付図 3-2 に例示した様に，上から順番にクラスの層が並んでいるものとする．付図 3-2 では，第 1 層には，独立クラス 3 個が並べられている．そして，第 1 層の下には第 2 層，第 3 層・・・と次々に層を設けている．

最初は２次元平面上にクラスは存在しない．そこで，トポロジカルソートの結果を，まず，第 1 層に左から右へと，1 個ずつ割り付ける．ただし，「割り付けようとしているクラスと同一層内には，関連で直接関係づけられたクラスは存在しない」ことが条件である．

新規に追加しようとしているクラスについて，候補位置と同一層に，上流側クラスが存在している場合には，ひとつ下の層に新規追加クラスの候補位置を移動する．そして，この位置から，再度，割り付け処理を継続する．結果的に，トポロジカルソート結果の順番で，左から右へ，かつ上から下へと，クラスが並ぶ．付図 3-2 の例では，直感的に配置した配置が，本付録 3 のアルゴリズムの結果に等しい．以上の分析から，1 対多関連のみを用いたクラス図では，以下が成立する．

（A）下流側クラスから上流側クラスへ向かう（直線で引いた）関連は，下流側クラス側から見ると，水平方向よりも上を向いている．下流側クラスを追加する際には，原則として，既存のクラスの位置を変更することなく，下部に追記できる．

（B）実際には，上流側クラスが存在しないのに，新規追加クラスを割り付けようとする恐れがある．この場合には，作成漏れのクラスを追加する必要がある．ただし，同一層内では，クラスの位置は自由である．同一層内では，関連による上流・下流関係が無いからである．従って，同一層内でクラスを平行移動してスペースを空けるだけで，作成漏れの上流側クラスを追加できる．

以上

著者略歴

金田 重郎（かねだ しげお）

1974 年 3 月京都大学工学部電気第二学科卒業, 1976 年 3 月京都大学大学院工学研究科電子工学専攻修士課程修了．1976 年 4 月日本電信電話公社・武蔵野電気通信研究所入所．1997 年 4 月同志社大学大学院総合政策科学研究科教授・同工学部教授．2020 年 3 月同志社大学退職．同志社大学名誉教授．