［研究論文］

工場モデルクラス図のための アプリケーション内業務処理プロセスの順位決定法

A Business Process Execution-ordering Method for Applications with Factory Model Class Diagrams

金田 重郎†
Shigeo KANEDA†

†同志社大学 名誉教授
†Honorary professor, Doshisha University

要旨

著者らは，概念データモデリングへの適用を狙いとして，「工場モデルクラス図」を提案している．工場モデルクラス図では，「独立クラス」から開始して，時間的進行に伴って，次々と起動される業務処理プロセスに対応したクラスを追記して行けば，クラス図が完成する．このため，工場モデルクラス図の作成に際しては，ターゲット AP を構成する業務処理プロセス相互の，時間的前後関係（半順序）を分析しておく必要がある．そこで，本論文では，業務処理プロセスの時間的前後関係の分析手法を提案する．提案手法は，独立クラスを業務処理プロセスのコンテキスト（文脈，背景）と見なす点に特徴がある．具体的には，以下の 3 ステップで分析を行う．(1) 最初に，ターゲット AP の仕様記述から「業務処理プロセス，及びそれに対応したクラス」を抽出し，併せて，独立クラスを抽出する．(2) 次に，業務処理プロセスを，独立クラス毎にグループ化して，グループ内部での時間的前後関係を決定する．(3) 最後に，グループ間に跨った時間的前後関係を明確化して，工場モデルクラス図を作成する．提案手法では，時間的前後関係の分析を 2 段階に分けたことにより，工場モデルクラス図を簡明かつシステマティックに作成できる．

Abstract

The authors have proposed “Factory Model Class Diagram (FMCD, hereinafter)” whose relationships are limited to one-to-many associations. The FMCD is created by starting from “Independent Classes” and adding the classes corresponding to the business processes activated one by one as time goes on. This means that FMCD creation requires the execution order of business processes in Application Program (AP, hereinafter). Therefore, this paper proposes a new method to acquire the business process execution order. Each business process has always independent classes in the upstream direction of the FMCD. The independent class serves as context of the business process in the downstream direction. Thus, the business processes can be divided into the groups each corresponding to one independent class. The proposed method has the following three steps: STEP1 acquires the class candidates and the business process candidates from the specification description of target AP, STEP2 groups the business processes having the same independent class into one group, and determines the execution order in each group. Also, STEP3 determines the execution order across the groups and creates the FMCD. The proposed method divides the execution-ordering into the two stages. This results in concise and systematical analysis of the business prosses execution order.

1. はじめに

UML クラス図は，概念データモデリングのための重要なツールである．著者らは，「関連」を 1 対多関連に限定した「工場モデルクラス図」を既に提案している[1][2]．工場モデルクラス図は，「関連」を「1 対多関連」に限定したクラス図である．「関連」の限定が，クラス図としての表現能力に影響を与えることはない．工場モデルクラス図の特徴として，「独立クラス」からスタートして，時間的進行に伴って，次々と起動される業務処理プロセスに対応したクラスを追記して行けば，クラス図を作成できる．従って，工場モデルクラス図を作成するためには，アプリケーション（以下「AP」と呼ぶ）の仕様記述から業務処理プロセスを抽出し，その時間的前後関係（半順序）を明確化しておく必要がある．しかし，工場モデルクラス図に適した，時間的前後関係の分析手法は知られていない．

そこで，本論文では，仕様記述から工場モデルクラス図を作成する手法を提案する．提案手法は，「独立クラス」に着目している点に特徴がある．工場モデルクラス図では，独立クラスから下流側クラスへの関係は，因果関係の連鎖である．全ての業務処理プロセスは，独立クラスを「コンテキスト（文脈，背景）」として持っている．業務処理プロセスは，独立クラス毎にグループ化できる．提案手法では，

［研究論文］
2022 年 9 月 30 日受付, 2022 年 12 月 13 日改訂, 2023 年 3 月 2 日受理
© 情報システム学会

この性質を利用して，業務処理プロセスの時間的前後関係の分析を簡明化する．

提案手法は，以下の 3 ステップから構成される．(STEP1) AP の仕様記述から，業務処理プロセスの候補，及びそれに対応したクラスの候補を抽出して，併せて，独立クラスを抽出する．(STEP2) 業務処理プロセスを，独立クラス毎にグループ化して，グループ内での時間的前後関係を決定する．(STEP3) グループ間に跨った（またがった）時間的前後関係を明確化して，工場モデルクラス図を作成する．提案手法では，時間的前後関係の分析を 2 段階に分割している．このため，一度に分析しなければならない業務処理プロセスの個数が減少して，見通しの良い時間的前後関係の分析が可能となる．

以下，第 2 章では工場モデルクラス図について解説する．第 3 章では，独立クラスとコンテキストについて論じる．第 4 章では提案手法を示す．第 5 章は本論文のまとめである．

2. 工場モデルクラス図

本章では，既存研究[1][2]で扱われた工場モデルクラス図について概説する．

2.1. 工場モデルクラス図について

「工場モデルクラス図」は，UML クラス図において，「関連」を 1 対多関連のみに限定したクラス図である．ここでいう「関連」には，「集約」，「コンポジション」を含める．「汎化」については，本論文では，議論から除外する．図 1 は，工場モデルクラス図の例である．関連の多重度については，「多重度＝1」側のみを「1」として表示している．但し，「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」であるような場合も包含する．図 1 のクラス図には，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．

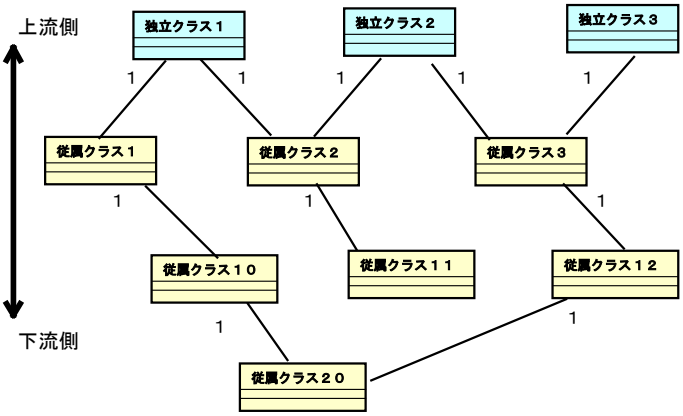

図 1　工場モデルクラス図の例[1]

工場モデルクラス図では，1 対多関連は，以下の 3 点を含意する[1].

【性質 1：時間的前後関係】 1 対多関連で関連付けられた 2 クラス間は，「多重度=1」側を前，「多重度=多」側を後とする時間的前後関係である．

【性質 2：因果関係】 1 対多関連は「因果関係」を含意しており，「多重度=1」側のクラスは，前提・材料・原因であり，反対側の「多重度＝多」側のクラスは，帰結・中間製品（製品）・結果である．

【性質 3：関数従属性】 1 対多関連は，「～が 1 つ決まると，～が 1 つ決まる」との意味であるから，関数従属性と類似している．この性質を，本論文では関数従属性と呼ぶ．

1 対多関連が持つ時間的前後関係で見ると，図 1 の工場モデルクラス図では，最上部の独立クラスからスタートして，「多重度=1」側から「多重度=多」側へと，時間的な進行に従って，業務処理プロセスが積み重ねられ，情報が流れて行く．「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」側のクラスを，「**下流側**」クラスと呼称する．図 1 全体の流れとして，独立クラスを初期材料として，時間的な推移に従って，次々とクラスが追加されて行く．この様子が，あたかも製品の加工組立工場のように見えるため，「**工場モデルクラス図**」と呼んでいる．

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．工場モデル

クラス図の性質として，1 つ以上の独立クラスが必ず存在する．独立クラスは，そこから上流にはクラスが存在しないクラスである．独立クラスでは，他クラスのインスタンス状態とは無関係に，インスタンスを追加できる．この独立クラスの性質は，ターゲット AP の仕様記述から独立クラスを抽出する際に活用できる．

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生成が，他クラスのインスタンスの状態に影響を受ける．1 つの従属クラスが，複数の上流側クラスを持つ場合には，複数の上流側クラスのインスタンスが揃って，初めて，下流側の業務処理プロセスを起動可能となる．

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，グラフを形成する．図 2 にそのイメージを示す．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．矢印の鏃（やじり）側が時間的に前である．本論文では，工場モデルクラス図として，この表現方法を利用する．矢印の向きと，情報の流れは逆であるので，注意されたい．工場モデルクラス図の持つグラフ構造については，上述の三つの性質に加えて，次の性質 4 が成立する[1].

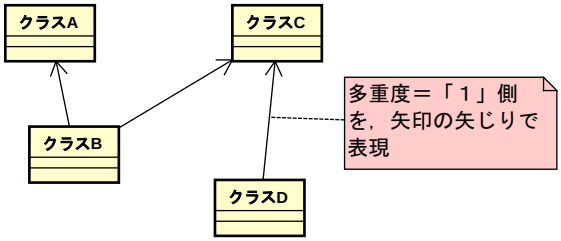

図 2　矢印による 1 対多関連の表現

【性質 4：DAG（Directed Acyclic Graph，有向非巡回グラフ）としての工場モデルクラス図】クラスをノード，関連をアークとする工場モデルクラス図のグラフ構造は，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．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 の推移簡約である．

「推移簡約」の意味を図 3 により説明する．図 3 左図では，クラス F のインスタンスが決まると，クラス G のインスタンスが決まる．更に，クラス G のインスタンスが決まると，クラス E のインスタンスが決まる．結果的に，クラス F のインスタンスが決まると，クラス E のインスタンスが決まる．

しかし，図 3 左図では，この 2 段階の関連による接続とは別に，クラス F からクラス E に向けて直接に関連が伸びている．明らかに，クラス F からクラス E へ直接伸びた関連は，冗長である．関係モデルにおける，推移的関数従属に該当する．従って，工場モデルクラス図では，図 3 右図のように，冗長な関連を消去して推移簡約化する必要がある．

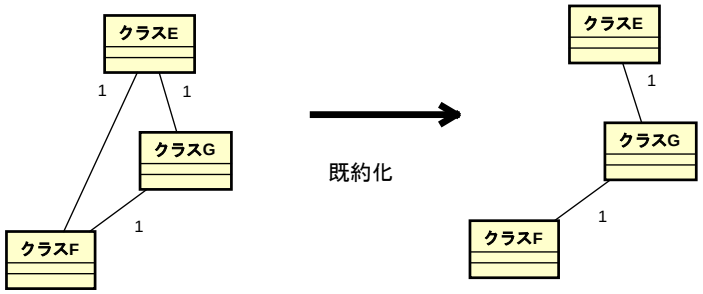

図 3　クラス図の推移簡約化

2.2. 工場モデルクラス図（テンプレート）

本論文では，仕様記述からの業務処理プロセスの抽出は，概念データモデリングへの利用を目的とする範囲に限定する．業務処理プロセスは，「クラス図に追加されるべきクラス」に 1 対 1 対応している必要がある．図 4 は，工場モデルクラス図の作成に用いるテンプレートである[1]．1 つの業務処理プロセスに対応して，1 つの従属クラスが規定されている．クラス図を描く際には，モデラーは，図 4 のよ

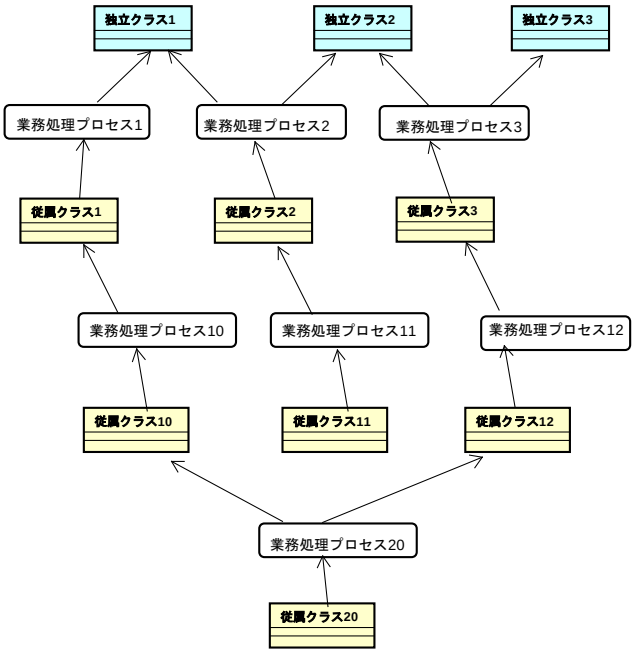

図 4　業務処理プロセスを追記した工場モデルクラス図（テンプレート）

うなテンプレートを思い浮かべている．業務処理プロセスは，対応するクラスと，事実上，一体である．

ターゲット AP の業務は，業務処理プロセスの連鎖として表現される．図 4 では，業務処理プロセスを，角の丸い長方形で表示している．処理内容は，「団地を登録する」「住戸の追加」，「家族の登録」といった表現で表示される．主語は，「システムは」「ユーザは」等であるが，通常，省略される．一方，時間的前後関係は，実線の矢印により表現している．矢印でリンクされている場合，上流側の業務処理プロセスが終了しないと，下流側の業務処理プロセスを起動できない．

なお，図 4 の工場モデルクラス図は，業務処理プロセスの連鎖であるため，一見すると，デマルコのデータフローダイアグラム（DFD, Data Flow Diagram）[3]に似ている．しかし，図 4 は DFD とは異なっている．図 4 では，業務処理プロセスが実行されると，その活動を記録する為に，当該業務処理プロセスに対応したクラスのインスタンスが生成される．図 4 の各業務処理プロセスのすぐ下に書かれているのが，対応しているクラスである．

これに対して，DFD では，「プロセス」（工場モデルクラス図における業務処理プロセスに相当）から出力されるデータは，当該「プロセス」により新規に生成されたデータとは限らない．1 つの帳票が，次々と属性値を追記されながら，DFD 上を流れて行ったりする．「プロセス」は，他「プロセス」が過去に生成したデータに対して，読み出し・修正・消去等を行う．

3. 独立クラスとコンテキスト

本章では，「独立クラス」に新しい解釈を加える．独立クラスから見て下流側に存在する従属クラス（及び，その従属クラスが対応する業務処理プロセス）は，コンテキスト（背景，文脈）として，「独立クラスを持っている」と解釈する．

3.1. 独立クラスと下流側クラスの関係性

本節では，「コンテキスト」を説明する．図 5 は，2 つの独立クラス（独立クラス A，独立クラス B）を持つ工場モデルクラス図である．例えば，従属クラス 1 は，独立クラス A を原因・原料・前提としており，従属クラス 1 のコンテキストとしては，独立クラス A が存在する．従属クラス 2 は，直接的に独立クラス A を原因・材料とはしていないが，間接的に原因・原料・前提として，独立クラス A に基づいている．同様にして，従属クラス 3，従属クラス 6，従属クラス 7 も，コンテキストに独立クラス A を持つ．これに対して，従属クラス 4 や従属クラス 5 は，独立クラス A との間に因果関係を持たない．

図 5 に示したように，独立クラス A の下流側にある全ての従属クラスは，独立クラス A と何らかの因果関係を持つ．これを，本論文では，「(独立クラス A の）コンテキストグループ」と呼ぶ．従属クラス 1，従属クラス 2，従属クラス 3，従属クラス 6，従属クラス 7 は，独立クラス A との因果関係を持つから，独立クラス A のコンテキストグループに含まれる．

1 つの従属クラスが，複数のコンテキストグループに属することもある．図 5 では，従属クラス 6，従属クラス 7 が複数のコンテキストグループに所属している．このような場合，図 6 に示したように，従属クラス 6 と従属クラス 7 の 2 つを別のコンテキストグループとすることも考えられる．新たなコンテキストを代表するクラスは，従属クラス 6 である．

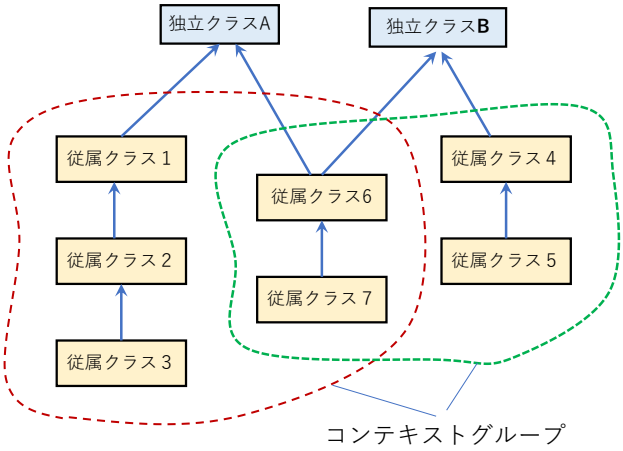

図 5　独立クラスとコンテキストグループ（I）

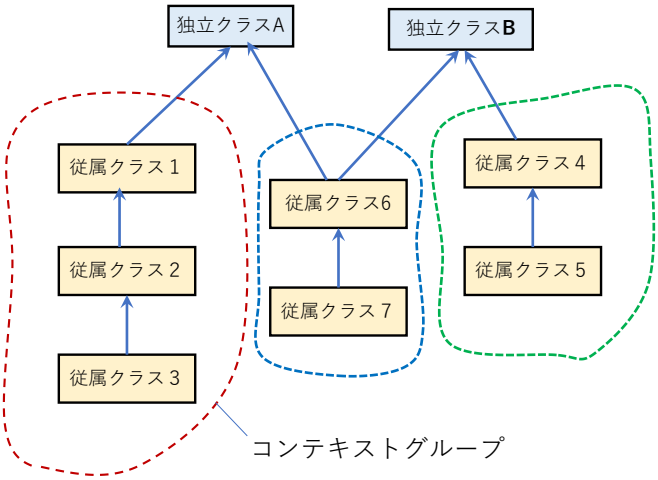

図 6　独立クラスとコンテキストグループ（II）

なお，図 5 では，クラス A とクラス B の 2 つの独立クラスから従属クラス 6 が作成されている．工場モデルクラス図では，「従属クラスのインスタンス」から「上流側クラスのインスタンス」にリンクが伸びる．この時，一度，リンクが張られると，上流側インスタンスの取り換えはできない．図 6 の例で言えば，従属クラス 6 のインスタンスを作成すると，そのインスタンスから，独立クラス A，及び独立クラス B のインスタンスへ伸びたリンクは後から修正できない．この性質は，コンテキストが変化しないことを保証する．

なお，図 7 に示すように，上流側にあるクラスのインスタンスの属性値を利用したいが，コンテキストとしての影響は切断したい場合も考えられる．この場合のクラス図としての実装方法は，次の 3.2 節で説明する．

3.2. コンテキストの継続と切断

本節では，コンテキストの継承を実装イメージで説明する．図 8 は，2 入力の業務処理プロセスに対応したクラス（「クラス C」）の実装イメージの例である．上流側には，2 つのクラス（「クラス A」，「クラス B」）がある．下流側のクラス C は，上流の 2 つのクラスの Object-ID を，属性値として持っている．このような構造を持つ工場モデルクラス図では下流側クラスは，直上のそれぞれの上流側クラスの Object-ID を持っているので，任意の従属クラスから，次々と上流側のクラスを辿り，独立クラスまで到達できる．

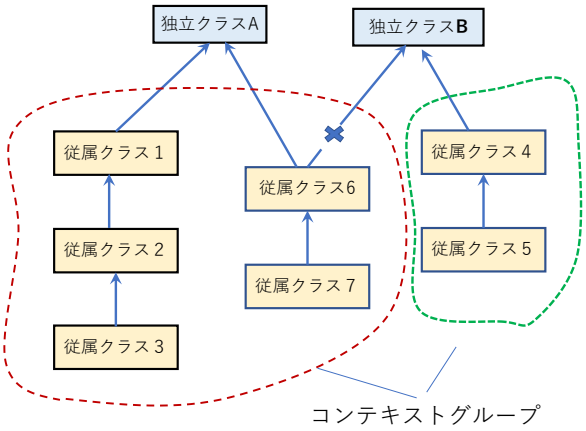

図 7　独立クラスとコンテキストグループ（III）

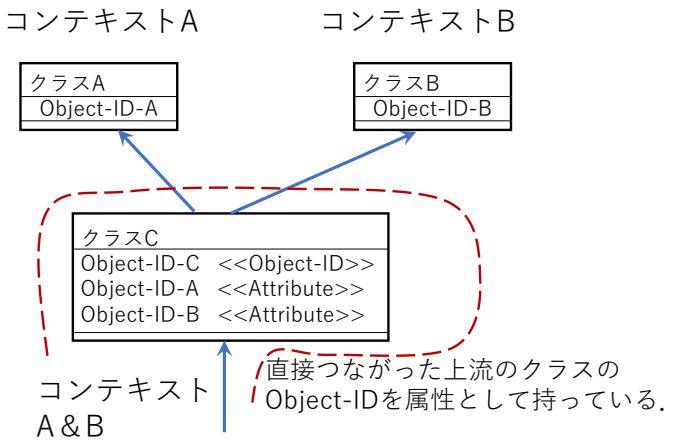

図 8　コンテキスト継承の実装例

一方，図 7 のように，一方の独立クラスのコンテキストを継承したいが，他方の独立クラスからは継承したくない場合を考える．但し，継承したくない独立クラスについては，継承はしたくないが，属性値を使いたいものとする．図 9 は，そのような場合の実装方法の 1 つである．クラス D は，クラス A のコンテキストを継承する．一方，クラス A とクラス B には関係性があるとする．この場合には，1 つの方法として，図 9 に示すように，クラス A とクラス B を，「関連のクラス化」であるクラス C を介して接続すれば良い．図 9 では，右側のクラス B のコンテキストは，クラス C には流れるが，そこで断絶している．クラス B のコンテキストが，あまり多くの従属クラス（業務処理プロセス）を持たない場合，図 9 の実装方法を採用して，コンテキストの及ぶ範囲を制限できる．

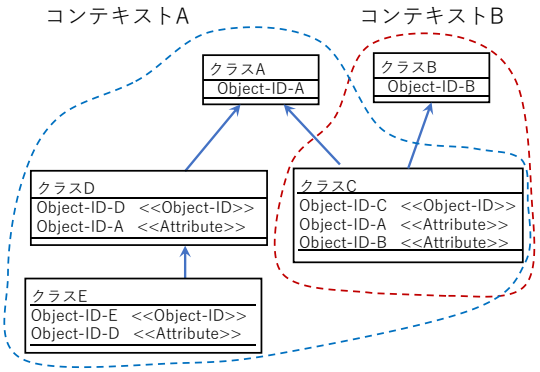

図 9　クラス B のコンテキストを継承しない時の例

独立クラスではなく，従属クラスをコンテキストとした方が良い例を図 10 に示す．図 10 は，受注業務を表現したクラス図である．顧客から注文があると，担当営業を 1 人決めて，受注をする．受注した

商品（種別）と個数の情報は，受注明細として作成している．ここでは，独立クラスは，「顧客」，「担当営業」，「商品」である．「顧客」，「担当営業」，「商品」のインスタンスは，他クラスのデータ状態に無関係に追加できるからである．図 10 には，受注クラスからスタートして，「受注明細」を通り，その後，在庫品割当，出庫等の業務処理プロセスを経由するのであろう情報の流れが存在する．そのコンテキストとしては，独立クラス「顧客」，「担当営業」ではなく，従属クラス「受注」が適切であると思われる．

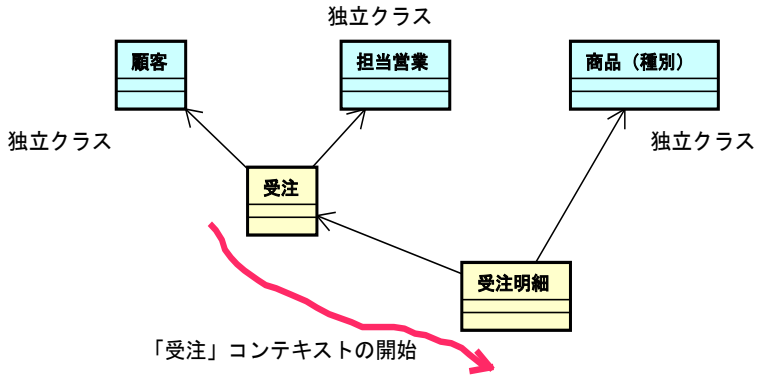

図 10　コンテキストの継承の例

4.　提案手法

本章では，提案手法について説明する．提案手法は，ターゲット AP の仕様記述から，「業務処理プロセスの候補」と「クラスの候補」とを抽出して，工場モデルクラス図を作成する．この一連の手続き自体は新しいものではない．提案手法のオリジナリティは，仕様記述から抽出された業務処理プロセス相互の時間的前後関係（半順序）を分析する際に，独立クラスを利用して，分析の効率化を実現している点にある．

4.1.　提案手法の原理

図 11（左図）は，従来手法における，時間的前後関係の分析イメージである．従来手法では，業務処理プロセス相互の時間的前後関係（半順序）の決定に際して，業務処理プロセス全体を，一度に分析する必要がある．分析対象となる業務処理プロセスの個数が多いため，煩雑な作業となる．

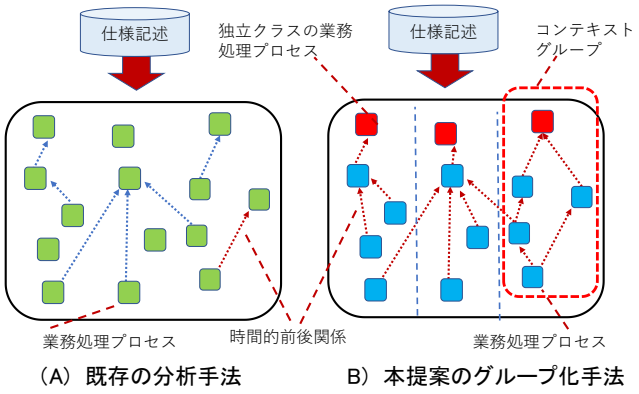

図 11　既存手法と提案手法に基づく時間的順序の決定

これに対して，図 11（右図）の提案手法では，業務処理プロセスの時間的前後関係を分析する前に，業務処理プロセスを，意味的に最も近い独立クラスを持つコンテキストグループに振り分けている．仕様記述から独立クラスを取り出すことは難しくない．

そして，全ての業務処理プロセスをコンテキストグループへ振り分けた後，時間的前後関係分析の第 1 段階として，コンテキストグループ内における業務処理プロセスの時間的前後関係を決定する．次に，第 2 段階として，コンテキストグループ間に跨った業務処理プロセスの時間的前後関係を分析する．結果的に，一度に順序付けを要求される業務処理プロセスの個数は減少する．時間的前後関係の分析が，従来手法よりも容易と考えられる．

そもそも，業務処理プロセス相互で，時間的順序関係（半順序）を付けることができることは，その

2 つの業務処理プロセスが意味的に近いものと考えられる．従って，独立クラス毎に業務処理プロセスをコンテキストグループに分割しても，半順序は，各コンテキストグループ内部で決定できる場合が多く，逆に，コンテキストグループを跨ぐ半順序関係は，因果関係が薄いだけに，半順序関係の個数自体が少ないことを期待している．

なお，前章で論じたように，何をコンテキストグループと見なすかは，主観的な側面を持つ．従って，実際の分析では，とりあえず，あるコンテキストグループ構成で分析を進めて，コンテキストグループ間に跨った順序関係を分析する段階で，コンテキストグループ境界の見直しと，業務処理プロセスの振り分け直しとが必要とされる場合もあり得る．

4.2. 用語と準備

提案手法の説明の前に，提案手法で使用される用語等について説明する．

【業務処理プロセスの表現方法】

提案手法では，最初にターゲット AP の仕様記述から，業務処理プロセスを抽出する．この際，業務処理プロセスを，どのように表現するかが問題となる．本論文では， UML ユースケース図のユースケースを想定する．即ち，英語の他動詞，あるいは目的語を取る動詞句のような構造を持つ単文である．動詞性名詞を用いた体言止めでも良い．例えば，「団地を登録する」「入居申請を受け付ける」「団地を登録」等の表現である．但し，多少の条件等の追記は許容することにする．主語は，「システムは」「ユーザは」であるが，省略される．

業務処理プロセスには，CRUD (Create, Read, Update, and Delete) の 4 種類がある．提案手法は，概念データモデリングへの適用を狙いとしている．パーシステント化すべき重要データの構造（クラス）を，次々にクラス図に追記して行く形で，工場モデルクラス図は作成される．実際の AP の処理としては，業務処理プロセスが実行されると，その活動記録として，インスタンスが生成されるイメージである．インスタンスの生成は，CRUD の「C」でのみ行われる．インスタンス生成には，当該業務処理に対応したクラス定義を必要とする．一方，CRUD の「RUD」では，業務処理プロセスの実行を記録する新たなインスタンスは生成されない．このため，クラス定義を必要としない．本論文のモデリングの範囲外となる．

【業務処理プロセスとクラス】

以上の議論から，業務処理プロセスは，常に，対応するクラスとペアを構成する．図 12 には，業務処理プロセスとクラスとの関係を示した．例えば，図 12 に対して「『クラス 3』の上流側に『業務処理プロセス 2』が存在する」との表現も可能である．ペアを構成しているので，本来は，「業務処理プロセス」ではなく，「業務処理プロセス，及びそれに対応したクラス」と表現するべきかも知れない．しかし，これでは煩雑である．本論文では，煩雑さを避けるため，誤解を招かない範囲で，「業務処理プロセス」と「クラス」とを使い分ける．

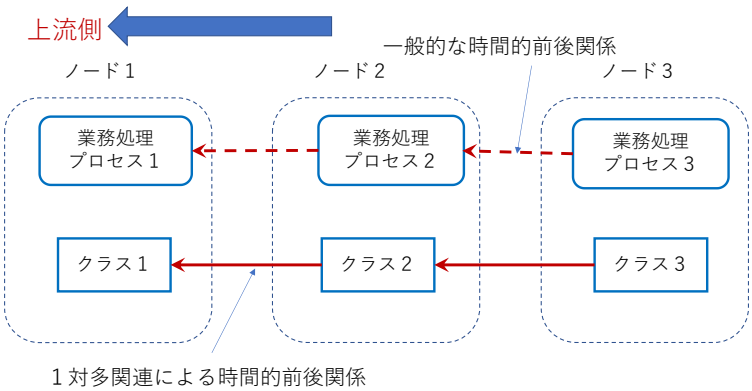

図 12　業務処理プロセスとクラス

【時間的前後関係】

提案手法では，仕様記述から業務処理プロセスとそれに対応したクラスを抽出し，ここから業務処理プロセス相互の時間的前後関係（半順序）を分析して，最終的には，工場モデルクラス図に変換する．提案手法では，時間的前後関係の分析を，STEP2 と STEP3 とで行う．但し，この 2 つで，利用する時間的前後関係の表現方法は異なる．

具体的には， STEP2 では，一般的な時間的前後関係を用いて分析し，STEP3 では，時間的前後関係を，1 対多関連を用いて表現する．ここで，一般的な時間的前後関係とは，通常の生活で経験する，「～

が終了すれば，～が起動可能となる」という因果関係である．一方，STEP3 では，1 対多関連を用いて，時間的前後関係を表現する．STEP2 から STEP3 の間で，時間的前後関係の表現方法が変化している．提案手法の適用に際しては，以下の点に注意するべきである．

（1） 1 つの業務処理プロセスが，上流側に，複数の業務処理プロセスを持つ場合には，上流側の全業務処理プロセスの処理が終了していないと，当該業務処理プロセスを起動できない．仕様作成に際して，注意が必要である．詳細は，4.5 節に示す．

（2）1 つの上流側業務処理プロセス（クラス）から，複数インスタンスのデータを取り出す業務処理プロセスを持つクラスを考える．この種のクラスを，既存研究[1]では，「**統計型クラス**」と呼んでいる．統計型クラスの場合は，一般的な時間的前後関係を用いて分析している STEP2 の段階では，問題なく表現できる．しかし，STEP3 の工場モデルクラス図では，正確には表現できない．クラス図上の注記等で対応する必要がある．詳細は 4.6 節で議論する．

（3）工場モデルクラス図では，上流側クラスから下流側クラスへ向けて，情報が流れている．逆方向の，下流側クラスから上流側クラスへのデータ伝搬は想定されていない．しかし，実際には，過去に生成されたクラス（上流側クラス）のインスタンスを，下流側クラスから，更新する処理が必要となる場合も想定される．既存研究[1] では，これを「**在庫型クラス**」と呼んだ．在庫型クラスは，STEP2 及び STEP 3 の両方で，正確には表現できない．下流側クラスから上流側クラスへ向けた情報の流れを表現できないからである．上流から下流へ流れる，通常の工場モデルクラスを描いておき，注記等で補う必要がある．詳細は 4.6 節で議論する．

4.3. 提案手法の詳細

本節では，提案手法を説明する．図 13 は，提案手法による工場モデルクラス図の作成過程である．提案手法は，大きく分けて，（STEP1）ターゲット AP の仕様記述の分析，（STEP2）コンテキストグループ内部での時間的前後関係の分析，及び（STEP3）コンテキストグループ間での時間的前後関係の分析，から構成される．併せて，STEP3 では，工場モデルクラス図が作成される．以下に，各 STEP について説明する．

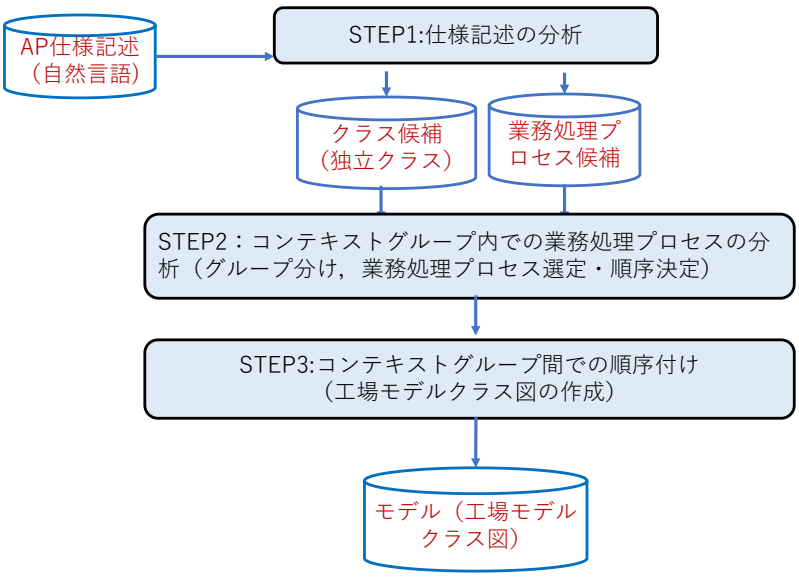

図 13　工場モデルクラス図の作成過程

STEP1：仕様記述の分析

STEP1 では，ターゲット AP の仕様記述から，クラスの候補（名詞），及び業務処理プロセスの候補を抽出する．クラスの候補からは独立クラスを取り出しておく．業務処理プロセスは（当該業務処理プロセスが出力するインスタンスを規定する）クラスとペアになる．例えば，仕様記述例に挙げた「団地を登録する」業務処理プロセスには，「団地」クラスが対応する．抽出すべき業務処理プロセスは，インスタンス生成を伴う場合のみである．

仕様記述から抽出されたクラスの中から，独立クラスを取り出すのは難しくない．独立クラスは，他クラスのインスタンス状態に無関係に，インスタンスを追加できる．例えば，「顧客」クラスは，他クラスには無関係に，新しい顧客を迎え入れることができる．逆に，「受注」クラスに関しては，誰からの受注なのか分からなくて済むことはあり得ないので，「受注」クラスは独立クラスではない．

STEP2：コンテキストグループ内での業務処理プロセスの分析

STEP2 は，2 つの処理から構成される．最初に，STEP1 で得られた「業務処理プロセス，及びそれに対応したクラス」を独立クラス毎にグループ分けする．各業務処理プロセスを，意味的に近い独立クラスを持つコンテキストグループに割り付ける．この場合，「業務処理プロセスには，少なくとも 1 つの独立クラスが必ず存在する」ことが理論的に保証されている必要がある．この点については，後ほど，確認する．

コンテキストグループへの振り分けが終了したら，次に，コンテキストグループ毎に，業務処理プロセス相互の時間的前後関係（半順序）を決める．注意が必要なのは，ここで利用する時間的前後関係である．STEP2 では，自然言語で記述された業務処理プロセスを対象とする．このため，我々が日常で意識している一般的な時間的前後関係により業務処理プロセスの順序を判断するしかない．以上が STEP2 の処理である．

ここで，業務処理プロセスと時間的前後関係が持つグラフ構造について確認する．業務処理プロセス，及び時間的前後関係は，業務処理プロセスをノード，時間的前後関係をアークと見なすと，グラフを構成する．もしアークがループすると，時間的前後関係から見て矛盾が生じる．何故なら，ループを辿ってどんどん過去（上流側）に進むと，突然未来になるからである．モデリングが正しく行われていれば，ループは存在しない．よって，以下の性質 5 が成立する．

【性質 5：業務処理プロセスと DAG】業務処理プロセスと時間的前後関係とからなる表現は，トポロジカルな構造として，DAG を構成する．業務処理プロセスと時間的前後関係に対しても，工場モデルクラス図と同様に，独立クラス（に対応する業務処理プロセス），従属クラス（に対応する業務処理プロセス），上流側，下流側等の用語を利用できる．DAG であるので，DAG の性質として，各業務処理プロセスの上流側には，少なくとも 1 つの，独立クラス（に相当する業務処理プロセス）が存在する．

性質 5 から，業務処理プロセスには，それを割り付け得る独立クラス（それに相当する業務処理プロセス）が必ず存在する．

STEP3：コンテキストグループ間での順序付け

最後に STEP3 では，コンテキストグループ間に跨る時間的前後関係を分析して，業務処理プロセス全体の時間的前後関係を確定する．併せて，コンテキストグループ内についても，時間的前後関係を見直す．得られた業務処理プロセスの順序に基づいて，最終的に工場モデルクラス図を作成する．

上記 STEP2 の結果は，STEP3 に継承される．性質 5 から，STEP2 で得られるデータ構造は，DAG である．一方，STEP3 のデータ構造は，工場モデルクラス図であるから，同じく DAG である．トポロジカルな構造が一致することから，STEP2 のデータ楮図をそのまま STEP3 に移管できる．

しかし，STEP2 では，一般的な時間的前後関係を用いて分析した．一方，STEP3 における時間的前後関係は，1 対多関連（関数従属性）で表現される．STEP2 における時間的前後関係であって，STEP3 において 1 対多関連にそのままでは変換できない場合がある．この問題は，4.2 節で論じた．詳細な説明は，4.5 節，4.6 節を参照願いたい．

以上，提案手法の各ステップを説明した．次節では，簡単なターゲット AP の仕様書に基づいて，上記の STEP1 から STEP3 について，具体例をもとに説明する．

4.4. 提案手法の実施例

本節では，ターゲット AP の仕様記述から，工場モデルクラス図を作成する過程を例示する．なお，仕様記述は，実在の公営住宅管理システムを参考にしているが，簡単化されている．仕様記述は，紙幅の都合で，付録に示す．

STEP1：仕様記述の分析

STEP1 では，ターゲット AP の仕様記述（付録に記載）から，クラス（名詞），及び業務処理プロセスを取り出す．併せて，独立クラスを抽出する．仕様書の「3.システムを用いた業務処理」の中に現れる，「仮入居申請」，「本入居申請」，「退去申請」，「駐車場利用申請」，「駐車場返納申請」等の申請業務をどのようなクラスで表現するかについては，設計者の裁量が大きい．本論文では，議論を簡単化するため，「仮入居申請」，「本入居申請」，「駐車場利用申請」では，そのままクラスの名称とした．処理内容記録のためのクラスのイメージである．

一方，CRUD の「RUD」処理は，抽出されない．具体的には，「退去申請」，「駐車場返納申請」は，インスタンスを消去するので含めない．「団地」，「住戸」，「契約者」，「同居家族」，「管理事務所」，「駐車場」，「駐車スペース」の各クラスのインスタンスに対する，値の読み取り処理，修正処理，消去処理も，同様の理由で，抽出対象には含めない．

以上の分析により，業務処理プロセスとクラスのペアが仕様記述から抽出される．得られたクラスは，「団地」，「住戸」，「契約者」，「同居家族」，「管理事務所」，「駐車場」，「駐車スペース」，「仮入居申請」，「本入居申請」，「駐車場利用申請」，「駐車場団地所属」「団地管理事務所所属」である．これらの中で，他クラスのインスタンスの存在に無関係に，自由にインスタンスを追加できるクラスが独立クラスであり，「団地」，「契約者」，「管理事務所」，「駐車場」の 4 つが該当する．

なお，仕様書では，駐車料金の概要をとりあえず計算する業務処理プロセスと，正式な借り受け開始日が入って利用料金を計算する業務処理プロセスとが区別されている．このため，「駐車場利用申請」については，「仮駐車場利用申請」と「本駐車場利用申請」の 2 つの業務処理プロセスに分けて考えることとした．得られる業務処理プロセスとしては，

・入居希望の人（契約者）を登録する．
・駐車場に駐車スペースを登録する．
・契約者からの本入居申請を受け付ける．
・・・・・

等がある．得られる業務処理プロセスは，表 1 の「業務処理プロセス」欄を参照されたい．但し，表 1 は，STEP2 の前半終了時の途中結果である．独立クラスとの関係，業務処理プロセスの順番は，STEP1 段階では定まっていない．なお，表 1 では，業務処理プロセスの内容表現として，並行した動作等に関する若干の表現を付加したものがある．

STEP2：コンテキストグループ内での業務処理プロセスの分析

STEP2 では，各業務処理プロセスを，該当するコンテキストグループに割り付け，その後，コンテキストグループ毎に，時間的前後関係を分析する．表 1 の「業務処理プロセス」欄は，4 つの独立クラスに割り付けられた，業務処理プロセスを示している．但し，表 1 では，業務処理プロセスの順序は与えられていない．

表 1．業務処理プロセスの抽出結果（STEP2 前半終了時点）

項目 ＼ 独立クラス	**業務処理プロセス** （新しくインスタンスを生成するプロセスに限定）	**クラス** （左記業務処理プロセスが生成するインスタンスを規定）
管理事務所	・新しい管理事務所を登録する． ・管理事務所を団地に関連付ける．	管理事務所 団地管理事務所所属
団地	・新しい団地を登録する． ・新しい住戸を登録する． ・住戸を指定して契約者からの仮入居申請を受け付ける． ・契約者からの本入居申請を受け付ける．	団地 住戸 仮入居申請 本入居申請
契約者	・入居希望の人（契約者）を登録する． ・同居家族を登録する．	契約者 同居家族
駐車場	・新しい駐車場を登録する． ・駐車場に駐車スペースを登録する． ・駐車場を団地に所属させる． ・仮駐車場利用申請を受け付けて，仮駐車スペース利用料金を計算する． ・本駐車場利用申請を受け付けて，契約者を駐車スペースに割り付ける．	駐車場 駐車スペース 駐車場団地所属 仮駐車場利用申請 本駐車場利用申請

表 1 で注目すべきは，業務処理プロセスの割り付けの直感的な分かり易さである．例えば，仕様書の中では，駐車場の借り受け処理は，住戸の借り受け処理の文脈で必要になるかも知れない．しかし，駐車場の借り受け（駐車場利用申請）と，住戸の借り受け（入居申請）は，背景となる独立クラスが明確に異なる．駐車場利用申請は，駐車場コンテキストグループに入っており，入居申請は，団地コンテキストグループに入っている．

各コンテキストグループに所属する業務処理プロセスが確定したら，次に，コンテキストグループ内部での業務処理プロセスの順序（半順序）を分析する．表 1 の「業務処理プロセス」欄では，コンテキストグループへの分類は終了している．しかし，時間的前後関係は確定していない．各業務処理プロセスに対応したクラスは，表 1 の右端の欄に付記した．

STEP2 の最終的な結果を，図 14 に示す．各コンテキストグループ内の業務処理プロセスは，上部から下部へと順序（時間的前後関係）を表す．時間的前後関係は，一般的な時間的前後関係である．例えば，図 14 の契約者コンテキストグループを見ると，1）最初に，契約者のインスタンスが追加され，2）次に，契約者の同居家族を登録する業務処理プロセスが走り，同居家族のインスタンスが生成される．「同居家族を登録する」業務処理プロセスは，「入居希望の人（契約者）を登録する」業務処理プロセスよりも後で実行することを矢印で示している．契約者が存在しないのに，その同居家族を登録するなど，考えられないからである．

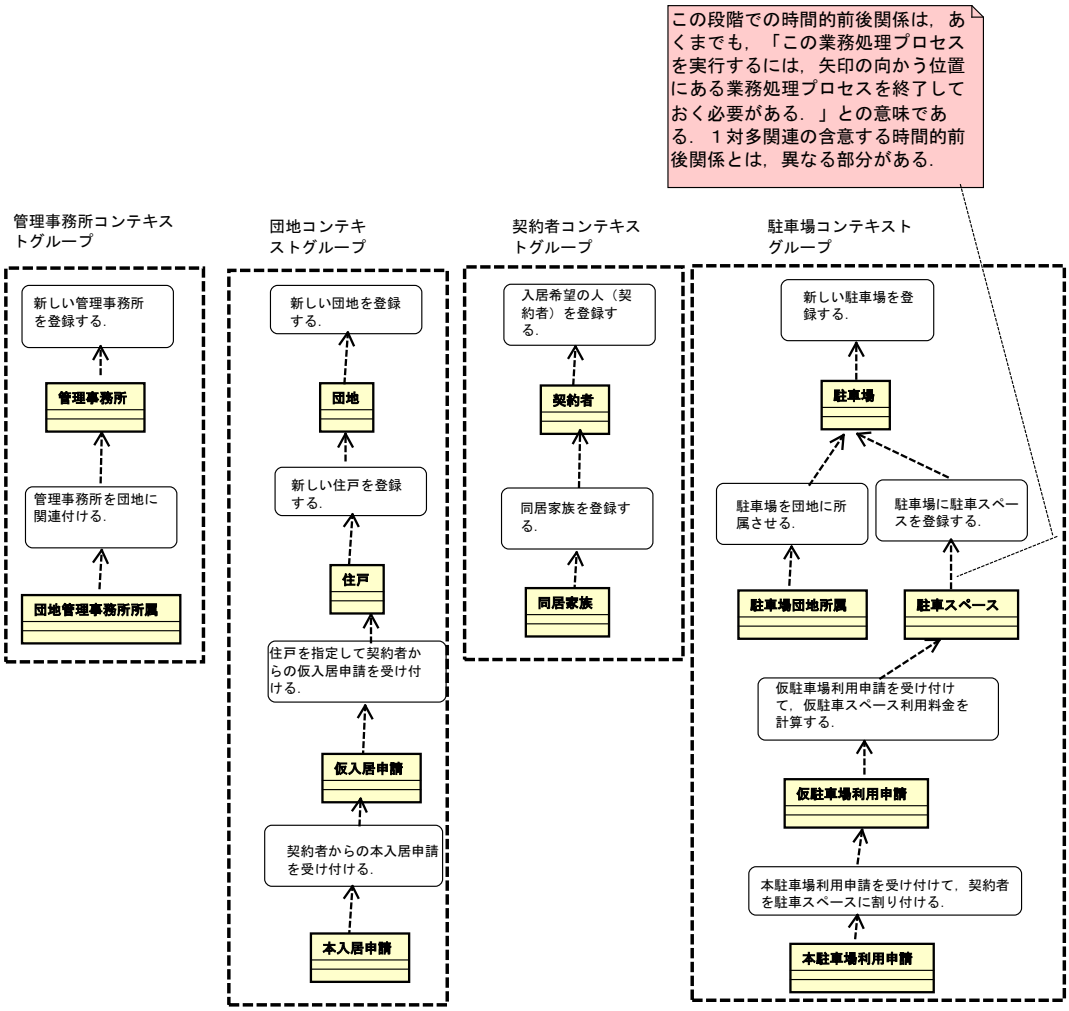

図 14　コンテキストグループ毎の順序付けが終了した状態（STEP2 終了時）

一方，駐車場コンテキストグループでは，時間的前後関係は，2 つに分岐している．新設の駐車場は，どこかの団地に所属させる必要があり，更に，その新設の駐車場には，駐車用のスペースを登録しておかないと，契約者の車を駐車することができない．「駐車場を団地に所属させる」「駐車場に駐車スペースを登録する」の 2 つの業務処理プロセスは，どちらか一方を必ず先に実行する必要がある訳ではない．但し，両方とも，駐車場の名称が，システムに登録されていなければ，処理は不可能である．このため，

両業務処理プロセスは，「駐車場を登録する」業務処理プロセスよりも後になる．

STEP3：コンテキストグループ間での順序付け

STEP3 は，工場モデルクラス図（テンプレート）を作成して，最終的なクラス図を得るステップである．図 14 を処理して図 15 を作成する．図 15 は，コンテキストグループ内とコンテキストグループ間での順序付けが終了した状態である．工場モデルクラス図（テンプレート）（図 4）に相当する．但し，独立クラスは，「(独立クラス) ～の登録」という業務処理プロセスにより表現している．図 15 における時間的前後関係は，1 対多関連である．

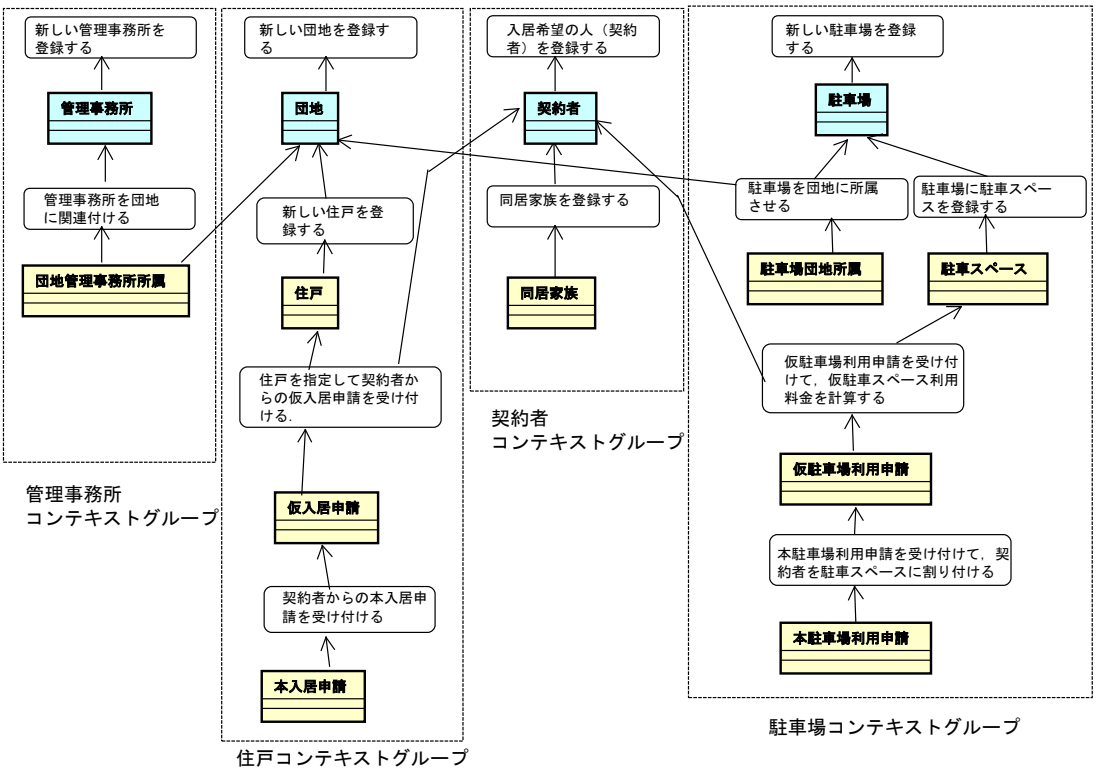

図 15　工場モデルクラス図（テンプレート）

図 15 において，団地コンテキストグループの，仮入居申請から出発している 1 対多関連に注意されたい．仮入居申請では，契約者の氏名，同居家族一覧，申請対象の住戸が必要である．しかし，仮入居申請から同居家族へは 1 対多関連は伸びていない．同居家族の 1 つ上流の，契約者に関連が伸びている．これは，同居家族クラスから複数のインスタンスのデータを取得することを，工場モデルクラス図の枠組みでは表現できないからである．図 16 は，最終的に得られたクラス図である．詳細な属性は省略している．

4.5. 起動条件の STEP2・STEP3 における相違点

本節では，1 つの業務処理プロセスが，複数の上流側業務処理プロセスを持つ場合の起動条件について，留意点を論じる．提案手法では，複数の上流側業務処理プロセスの全てが処理済みとならないと，当該業務処理プロセスを起動できない．

図 17 は，STEP2 の処理構造を前提とした，2 個の上流側業務処理プロセスを持つ，業務処理プロセスの例である ．但し，業務処理プロセスとそれに対応したクラスとは，まとめて表現した．業務処理プロセス 4 に着目すると，上流側に 2 個の業務処理プロセスを持っている．しかし，これは「業務処理プロセス 4 を起動するためには，業務処理プロセス 2 と業務処理プロセス 3 とが，両方とも実行済みであること」を必ずしも意味しない．業務処理プロセス 2 と業務処理プロセス 3 の少なくとも一方が実行済みであることが，業務処理プロセス 4 の起動条件であっても良い．STEP2 の範囲内では，一般的な時間的前後関係でモデルを記述できるため，仕様記述上の起動条件を自由に設定できる．

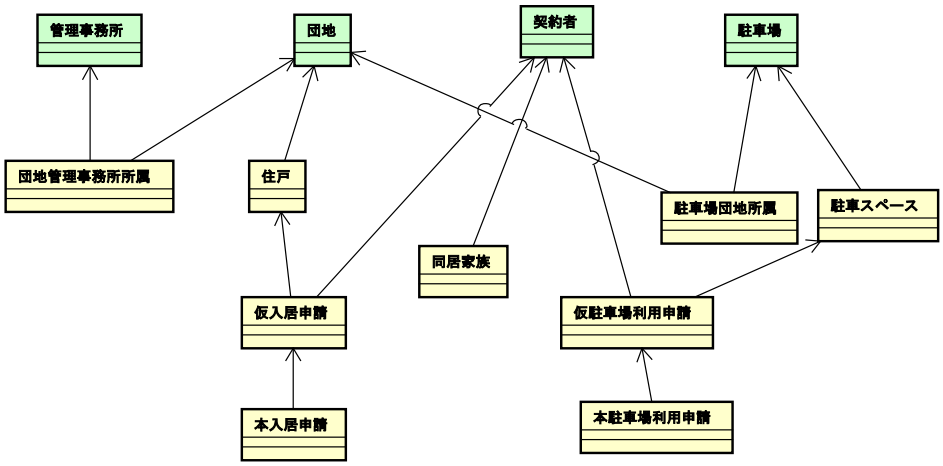

図 16　最終的に得られた工場モデルクラス図

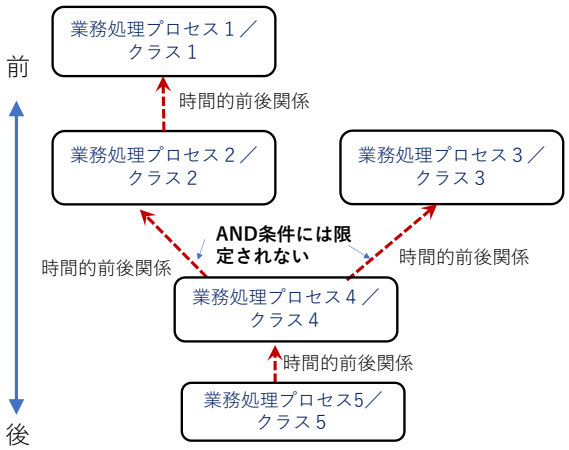

図 17　時間的前後関係（STEP2 の形式，STEP3 では扱えない）

これに対して，図 18 は，STEP3 の場合を示す．この場合には，業務処理プロセス 2 と業務処理プロセス 3 との両方の実行が終わっている時にのみ，業務処理プロセス 4 を起動できる．STEP3 では，時間的前後関係は，１対多関連で記述され，複数の上流側クラスを持つ場合，下流側クラスから上流側クラスに伸びた 1 対多関連は AND 条件である．

上の議論から，STEP2 だけを見れば，起動条件の設定は自由である．しかし，最終的な目標が工場モデルクラス図であるため，１つの業務処理プロセスが，複数の上流側業務処理プロセスを持つ場合，上流側の全ての業務処理プロセスの処理が終了して，初めて，業務処理プロセスが起動可能となる．

4.6.　「１対多関連」による時間的前後関係の表現

工場モデルクラス図では，クラス間の時間的前後関係を１対多関連で表現する．このため，重要な業務処理プロセスであるにも関わらず，工場モデルクラス図では，正確には表現できない場合がある．「在庫型クラス」と「統計型クラス」である．これらについては，既に，既存研究[1]で論じている．但し，STEP2 から STEP3 へ進む際の，時間的前後関係の表現手段の切り替えに関係した問題なので，本論文でも再度説明する．

【在庫型クラス】

「在庫型クラス」について説明する．在庫型クラスは，「下流側のクラス（業務処理プロセス）が，上流側クラスの既設インスタンスの属性値を更新する」クラスである．本来，工場モデルクラス図の情報の流れは，「上流側クラスから下流側クラスへ」の一方向である．しかし，在庫型クラスでは，逆の「下流側クラスから上流側クラスへ」の流れが存在する．工場モデルクラス図では，この逆方向の流れは，表現できない．クラス図に注記する等の対策を必要とする．

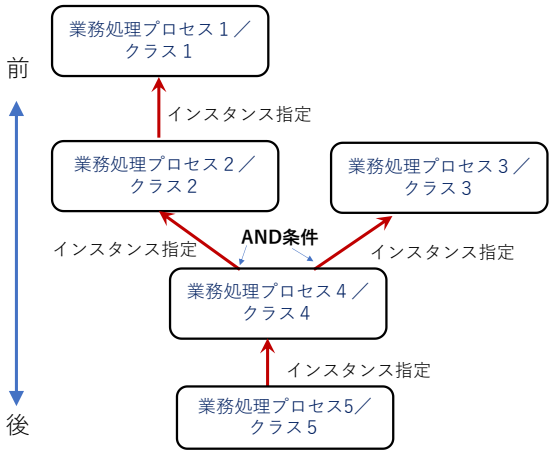

図 18　時間的前後関係（STEP3 の形式）

具体例で説明する．図 19 は在庫型クラスの簡単な例である．「在庫クラス」は，当該商品の在庫数・引当済数を保持している．但し，在庫数等は，倉庫毎に管理されており，倉庫横断的に集計することは考えていない．

図 19 の動きを確認する．ある時点において，在庫クラスのインスタンスが作られたとする．その時点での，商品の在庫数と引当済数が，当該インスタンスの属性値として記憶される．その後，受注明細が到着する．「在庫引当クラス」は，在庫数を減算し，引当済数を加算する必要がある．そこで，在庫引当クラス対応の業務処理プロセスが起動され，在庫引当クラスのインスタンスを作成する．そのためには，現状の在庫数・引当済数を在庫クラスのインスタンスから取得する必要がある．これは，上流（在庫クラス）から下流（在庫引当クラス）への，自然な情報の流れである．

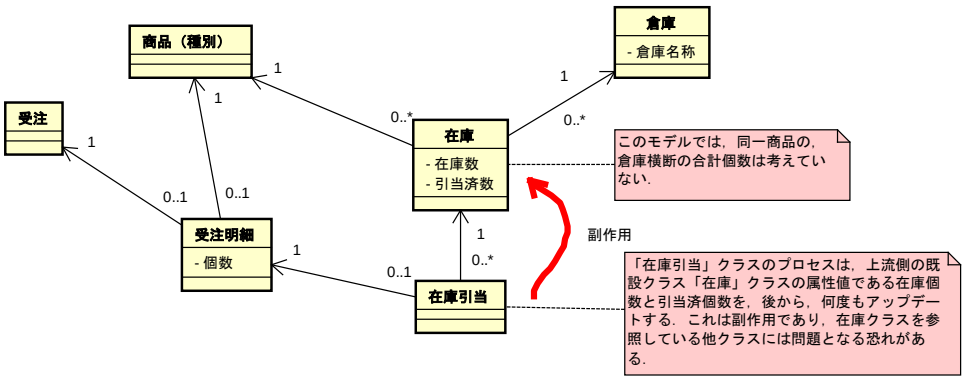

図 19　「在庫型クラス」の簡単な例（STEP3 の表記方法による）

次に，在庫引当クラスは，取得した在庫数等に基づいて，新しい在庫数・引当済数を計算して，元の在庫クラスのインスタンスに書き戻す．これは，通常とは逆方向の，下流から上流へ向かう情報の流れである．在庫引当クラスは，２方向の情報の流れを，持っている．

上記の逆方向の情報の流れは，STEP3 の工場モデルクラス図では，正確には表現できない．STEP2 の業務処理プロセスとその時間的前後関係から成る処理構造においても，時間的前後関係を半順序としている．逆方向の情報の流れは，表現できない．STEP2，STEP3 のいずれの場合でも，逆方向の情報の流れは，正確には表現できない．注記等により，逆方向の情報の流れの存在を示す必要がある．

【統計型クラス】

次に「統計型クラス」について説明する．統計型クラスは，１つの上流側クラスから，複数インスタンスのデータを取り込む点に特徴がある．STEP2 の「業務処理プロセスと一般的な時間的前後関係による表現」を用いた場合には，この統計型は自然に表現できる．下流側から上流側へ，矢印（破線）を引いて，時間的前後関係を表現すれば良い．複数インスタンスのデータの流れを，一つの矢印（破線）で示す事ができる．

しかし，STEP3 の 1 対多関連による時間的前後関係の表現では，問題が発生する．具体的な例を用いて説明したい．図 20 は，統計型クラスの例である．関連の矢印は，1 対多関連である．図 20 の仮入居申請クラスに着目する．仕様上，仮入居申請には，「契約者」「住戸」のデータに加えて，「同居家族」に関するデータが必要であると仮定する．STEP2 の一般的な時間的前後関係であれば，仮入居申請クラスから，住戸クラス及び同居家族クラスに，矢印を延ばせば良い．しかし，この要請は，STEP3 の工場モデルクラス図の枠組みでは，表現できない．1 対多関連は，家族 1 人分のデータしか，指定できないからである．

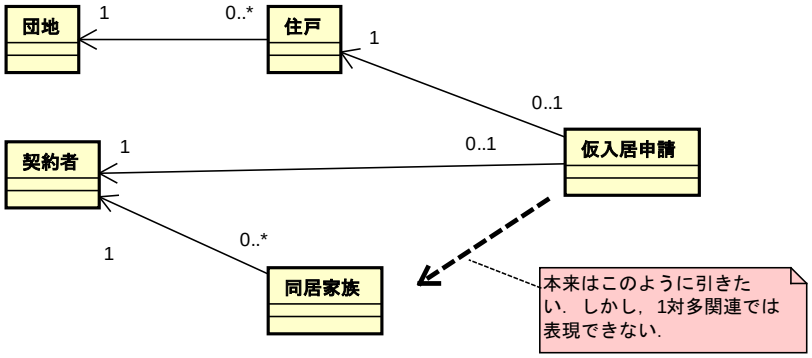

図 20　「統計型クラス」の簡単な例（STEP3 の表記方法による）

図 20 の例では，モデラーは，仮入居申請クラスから同居家族クラスへ伸びるべき 1 対多関連が描けないことを認識した上で，同居家族クラスから見て 1 つ上流側にある，契約者クラスへ，1 対多関連を伸ばした．この関連があれば，仮入居申請から同居家族クラスを探すことは可能となる．但し，上記の契約者クラスが持っている時間的位置は，本来の上流側クラスである同居家族クラスより前である．図 20 では，同居家族のデータが入力済みか否かを無視して，制御を仮入居申請クラスに渡すことが可能となる．アプリケーションの時間的前後関係としては，誤りの恐れがある．この点は，実装上は意識せねばならない．

4.7. 議論

関連研究として，手島歩三の概念データモデリング[4]について触れておく．手島の概念データモデリングでは，「静的モデル」，「動的モデル」，「組織間連携モデル」を用いてモデリングする．手島のアプローチでは，静的モデルにおいてエンティティ（クラス）を抽出する．そして，このエンティティが，動的モデルの中を流れて工程を重ね，その工程を通じて，属性値が，書き込まれたり，更新されたりする．動的モデルは，最後に，組織間連携モデルにおいて，組織マップの上に展開される．

組織間連携モデルは，DFD に似ている．モデルを記述していると，「1 つのエンティティが，名前を変えながら，組織の間を流れて行く」ように記述したくなる．例えば，「受注品」が，業務の進行に伴って，「受注品」→「在庫割当済品」→「出庫品」→「発送品」と名称変更されるイメージである．ドメインエキスパートから見た「名称の自然さ」を重要視するからである．しかし，結果として，エンティティの名称が動的に変化するので，オブジェクト指向の思想とは相入れない．このため，固定したエンティティ名称を利用することになる．

組織間連携モデルにおける情報の流れは，ひとつの文脈に基づいた，業務処理の連鎖のように見える．それが，鳥瞰的な広い視野を提供する．しかし，情報の流れのスタート時点におけるエンティティ名のまま，一連の処理を表現することには，無理があるのかも知れない．

これに対して，本論文では，最初に，文脈を明示した上で，文脈を継承するクラスを次々と追加する．クラスの名称は，処理の進行につれて，自然に変化し，オブジェクト指向の思想とも合致する．

また，椿正明は，情報システムの処理が，加工組立工場のようになるとした[5]．本論文の性質 5 が示す所によれば，加工組立工場のようになるのは，「業務処理プロセス，及びそれに対応したクラス」を構成単位として，ターゲット AP の処理を表現したことに起因する．「～を～する」という単文形式を用いて，処理内容を記述することは，情報処理技術の仕様記述では，ありふれたことである．モデラーは，汎用性があると考えて，そのような表現形式を採用している．しかし，性質 5 は，単文形式の表現では，汎用性を完全には担保できていない可能性を示す．「～を～する」パタンだけで，あらゆるアプリケーションの処理を表現することには，無理があるものと思われる．

著者らは，既存研究[6][7]において，クラス図の構造が英語の単文の認知構造に等しいことを示してい

る．その意味では，作られるものがクラス図なので，単文を用いてターゲット AP 仕様書から情報を得ることは，知識表現の粒度から見て，正しい選択と考える．しかし，単文はコンテクストフリーな表現である．業務処理内容を表現できても，業務処理の前後関係は，表現できない筈である．仕様記述から抽出されたコンテクストフリーの単文表現のみからでは，業務処理プロセスの実行順序を判定する事は難しい．モデラーは，ターゲット AP の仕様記述を十分に読み込んで，必要なら記述を補って，ターゲット AP の動的振る舞いを理解すべきであると考えられる．それが，提案手法の適用に際しての，暗黙の前提条件ではないだろうか.

5. おわりに

本論文では，ターゲット AP の仕様記述から，工場モデルクラス図を作成する手法を提案した．提案手法のオリジナリティは, 独立クラスを, その下流に存在する従属クラスのコンテキスト（背景・文脈）と見なす，新しい解釈を導入した点にある．仕様記述から抽出された業務処理プロセスは，独立クラス毎にグループ化できる．このグループを，独立クラスに対する「コンテキストグループ」と呼ぶ.

独立クラスは，仕様記述から容易に抽出できる．そこで，提案手法では，先ず，仕様記述から抽出した業務処理プロセスを，コンテキストグループに振り分ける．これにより，業務処理プロセス相互の時間的前後関係（半順序）の分析を，1）コンテキストグループ内部の分析と，2）コンテキストグループ間に跨った（またがった）分析との，2 段階に分割できる．その結果，効率的に工場モデルクラス図を作成できる.

但し，どのような業務処理プロセスの仕様でも，工場モデルクラス図に変換できるわけではない．以下の点に留意すべきである.

・第 1 の留意点は, 業務処理プロセスが, 複数の上流側業務処理プロセスを持つ場合の起動条件である．この種の業務処理プロセスは，全ての上流側業務処理プロセスが終了して，初めて，当該業務処理プロセスを起動できる仕様のみが許される.

・第２の留意点は，業務処理プロセスが，上流から下流に向けた情報の流れとは別に，下流から上流に向けた情報の流れを持つ場合である．この種の業務処理プロセスを「在庫型」と呼ぶ．下流から上流への情報の流れがあるので，在庫型の仕様は，工場モデルクラス図では正確には表現できない．クラス図上は，注記等として表現するべきと考える.

・第３の留意点は，１つの上流側業務処理プロセス（クラス）から，複数インスタンスを取得する業務処理プロセスの扱いである．この種の業務処理プロセスを「統計型」と呼ぶ．工場モデルクラス図では，統計型の仕様は，正確には表現できない．クラス図上の注記等で対応する必要が生じる.

本論文の提案手法に関する，未検討の課題としては，ターゲットシステムの規模が大きくなった場合における，提案手法の有効性がある．更なる評価が必要と考える.

参考文献

[1] 金田重郎, “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,” 情報システム学会論文誌, Vol.18, No.1, pp.1-21, 2022 年 9 月, https://www.issj.net/journal/jissj/index.html,（2022 年 9 月 30 日確認）.

[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. https://www.issj.net/journal/jissj/index.html,（2022 年 9 月 30 日確認）.

[3] トム・デマルコ, “構造化分析とシステム仕様,” 日経 BP 出版センター, 1994 年 9 月.

[4] 手島歩三, 松井洋満他, “働く人の心をつなぐ情報技術―概念データモデルの設計,” 白桃書房, 2011 年 6 月.

[5] 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.

[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ン―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 年 7 月, http://doi.org/10.14923/transinfj.2014JDP7103,（2022 年 9 月 30 日確認）.

[7] 金田重郎, “認知文法に基づくオブジェクト指向の理解,” 同志社大学政策学会, 同志社政策科学研究, Vol.13, No.2, pp.21-45, 2012 年 3 月, http://doi.org/10.14988/pa.2017.0000012727,（2022 年 9 月 30 日確認）.

付録　仕様記述例

公営住宅管理システム（仕様書）

1. システムの目的

A 県が保有する公営住宅（賃貸）の各住戸について，契約者の入居申請処理，退去申請処理，同居家族の登録管理，賃借料の計算など，を行う．

2. 用語

【管理事務所】A 県の公営住宅の管理を担当する事務所である．A 県の内部組織である．各公営住宅には 1 つの管理事務所が指定され，1 つの管理事務所が複数の公営住宅を管理することがある．
【団地】A 県は公営住宅を保有しており，公営住宅は「団地」と呼ばれ，団地は複数存在し，名称で区別される．団地は，複数の「住戸」を持ち，住戸は棟番号と住戸番号で識別される．全ての住戸は，賃貸である．
【契約者】住戸に入居している人（自然人のみ）を「契約者」と呼ぶ．契約者以外の同居する家族を「同居家族」と呼ぶ．
【駐車場】A 県は，団地とは別に複数の駐車場を持っている．駐車場も名称で区別され，駐車場は，番号で区別される複数の「駐車スペース」を持つ．駐車場は，最初から特定の団地に所属するのではなく，駐車場の完成後に，いずれかの団地に割り付けられる．1 つの団地が複数の駐車場を持つことがある．但し，1 つの駐車場が複数の団地に所属することはない．
【賃貸料】住戸の賃貸料は，各住戸に付与されたランクと契約者の所得，同居家族の人数により決まる．加えて，駐車場を契約している場合には，駐車場利用料が加算される．

3. システムを用いた業務処理

【住戸管理】入居申請者が希望する住戸を，団地名，棟番号，住戸番号によって識別して，最初に「仮入居申請」を提出する．このため，指定された団地について，希望条件に合致する住戸の空き状態を検索できる．仮入居申請では，契約者本人の情報だけではなく，同居家族に関する情報を登録する．この際，契約者の所得証明を添付する必要がある．仮入居申請があると，月毎の概算賃貸料を計算する．この概算賃貸料を契約者が了解した場合には，「本入居申請」に移行する．実際の入居日を考慮した，確定した賃貸料を再計算して，入居時の支払金額を確定する．入居後も，同居家族は追加，削除が生じうる．同居家族の増減があると，賃貸料は再計算される．契約者が住戸を退去する場合には，「退去申請」を行う．退去時には，賃貸料の清算分があれば，計算する．
【駐車場管理】契約者は，1 つの駐車スペースまでに限り，駐車スペースを賃借できる．指定された駐車場について，希望条件に合致する駐車スペースの空き状態を検索できる．駐車場スペースを利用するには，希望する駐車場の名称，及び駐車スペースの番号を指定して，「駐車場利用申請」を行う．住戸の賃貸料に，駐車場利用料金が加算される．駐車スペースを返却するには，「駐車場返納申請」を行う．賃貸料の清算部分が計算される．なお，駐車場利用料金は，賃貸料に上乗せされる．
【公営住宅管理】団地・管理事務所・駐車場の新規追加・修正・削除を行う．また，駐車場・管理事務所を団地に割り付ける．駐車場に所属する駐車スペースに契約者を追加・修正・削除する．団地に所属する住戸に契約者の追加・修正・削除を行う．

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．