情報処理学会論文誌 Vol. 50 No. 2 675–689 (Feb. 2009)

概念データモデリングとプロブレムフレームを用いた情報システム実装手法

吉田和正[†1] 金田重郎[†1] 芳賀博英[†1]

情報システム設計における要求分析フェーズにおいて，ユースケースのように，情報システムの機能を最初に記述するアプローチを採用すると，対象業務が持つ問題点が残置されたり，業務内容変更に対応困難なシステムが構築されたりしがちである．特定非営利法人・技術データ管理支援協会の概念データモデリング（CDM）は，このような問題点を回避するために，『業務の中のあるべきデータの流れ』を明確化する手法として知られる．しかし，得られた CDM モデルを実装レベルに変換する方法は知られておらず，CDM と実装の間にはギャップがある．そこで，本論文では，CDM モデルから，実装レベルの機能仕様を導く手法を提案する．具体的には，実装に向けた要素のうち機能に関する要求と仕様の分析にジャクソンのプロブレムフレームを用いることで，概念データモデリングでとらえた業務モデルから情報システムとして実装すべき機能を導出する．また，制御手順の導出に際しては，2 段階のシーケンス図を作成して対応する．提案手法の有効性を確認するため，ある自治体の市民税業務処理への適用実験を行い，機能レベルの実装モデルを導出できることを確認した．

Proposal of Information System Implementation Approach Using Conceptual Data Modeling and Problem Frames

Kazumasa Yoshida,[†1] Shigeo Kaneda[†1] and Hirohide Haga[†1]

Concept Data Modeling (CDM) of MASP (Manufacturing Architecture for Series Products Association) is a well-known requirement analysis method for the development project of large scale information systems and has been already applied for many huge software development projects. However, the CDM is just a model for business analysis among the stakeholders. The application system implementation process is not shown for the CDM. To resolve this problem, this paper proposes a new approach to provide an implementation model such as UMLs and a necessary function patterns from the CDM by using Jackson's Problem Frames. The proposed method was applied to a taxation business process in tax for citizens. The method provided functional specifications effectively for the tax operation business.

1. はじめに

情報システム開発分野では，従来，現場担当者からのヒアリングによって業務フローを分析し，帳票などのデータ項目から ER 図を作成して情報システムを構築する手法がしばしば用いられてきた．しかし近年，このアプローチには「開発されたシステムが改造に弱い」「業務全体での導入効果が乏しい」などの課題が指摘されている[1)]．

上記の課題を解決するアプローチとして，概念データモデリング（Conceptual Data Modeling，以下 CDM）が知られている．なかでも，特定非営利法人技術データ管理支援協会（MASP）[3)] は，対象世界の「もの」とそれに変化を起こす「こと」に注目した，独自の CDM 手法を提案している[4)–6)]．これにより，業務全体のデータ整合性と，改造が容易な情報システムを構築できるとしている[5)]．MASP の CDM については，KDDI[1)]，JFE スチール[7)] などの成功例が報告されている．

CDM では，対象ビジネスに関する価値観を捨象し，現実世界をあるがままにデータとして写し取る．ただし，CDM で表現されているのは，概念的な上位ビジネス記述であり，そのままでは，情報システムの詳細仕様をそこから導くことはできない．すなわち，CDM・実装間にはギャップが存在する．

そこで本論文では，CDM のモデルでとらえた「もの」「こと」を，情報システムの実装へつなぐ手法を提案する．具体的には，ビジネスにおける情報システムに必要な要素の抽出方法と，情報システムの構築に必要な概念クラスや機能種別を導く方法について述べる．情報システムに関するモデリングには UML[10)] とジャクソンのプロブレムフレーム[11)] を採用する．

以下 2 章では既存情報システム開発手法の問題点について述べる．3 章では CDM とプロブレムフレームについて述べる．4 章では CDM とプロブレムフレームを用いた分析手法を提案する．5 章では適用事例を示す．6 章では本提案手法の効果について考察する．7 章では，関連手法との比較を行う．8 章はまとめである．

†1 同志社大学大学院工学研究科
The Graduated School of Engineering, Doshisha University

2. 現状の課題

情報システム開発分野では，従来，情報システムの機能仕様を導くために，ユーザへのヒアリングによって対象業務を分析した業務フローを作成した後，UML のユースケース図や概念レベル・実装レベルのクラス図などを導く手法が多用されてきた．システムの構造が複雑化する場合には，必要に応じて別途実装に直接必要な ER 図やデータフローダイアグラムを導く場合も存在する．

しかし近年，この種のアプローチの課題として，「上流からの仕様変更に弱い」「現場の要求は満たすものの，設備やシステムに対する投資効果や，業務全体の効率化への寄与が希薄である」などが指摘されている．そして，このような効果にとどまる原因として，ビジネス側からは「ビジネス側のニーズが開発者に正しく伝えられているかを確認できない」，開発者側からは「ビジネスやユーザのニーズが曖昧なままに開発が進み，ニーズが明確になるころにはシステムの身動きがとれなくなっている」などの指摘がある．

上記の問題の開発者側からの解決案として，アジャイル開発プロセスや MVC フレームワークを通して，ユーザやビジネス側に早期のプロトタイプレビューをフィードバックし，双方のギャップを埋めようというアプローチが存在する．しかしながら，これらのプロセスの中で用いられるモデルは実装主体のレベルで記述されていることが多く，あくまで開発者が読むためのモデルが主流である．

一方でビジネス側からのアプローチとして，経済産業省を中心に EA（Enterprise Architecture）への関心が高まっている．EA は企業（Enterprise）の活動全体を様々な観点から記述・統合し，企業活動の全体像を明らかにすることで，それを支援する情報システムの明確化を目的とする．EA は**図 1** に示す 4 つの階層からなる．

EA の視点から見ると，従来の開発手法には，ビジネスの構造から情報システムを適用するまでの工程に問題があることになる．たとえば，現場ヒアリングからユースケース図を作成し，その中に現れる名詞と動詞からデータベースや画面の設計を行う方法がある．しかし名詞と動詞などの言語情報は必ずしも情報システム上に実装されるデータとは限らず，対象のビジネスのある領域を表現しているにすぎない可能性がある．

ヒアリングや業務分析から得た情報をそのまま実装するのではなく，この情報から対象ビジネスの本質を表現するデータ構造を吟味する工程が必要である．この工程を欠いたままに情報システム開発を行うと，ビジネスレベルの変更を情報システムへ反映することが難しくなる．EA 視点からすると，ビジネスと実装との間をうまくつなぐモデル・分析手法が必要である．

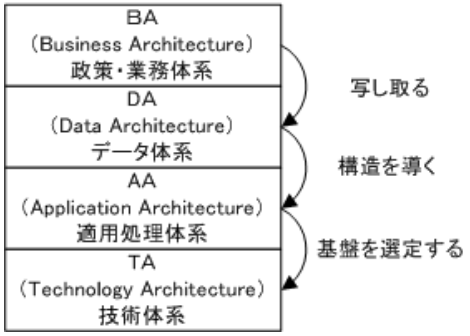


図 1　EA（Enterprise Architecture）の構造
Fig. 1　Structure of EA (Enterprise Architecture).

3. アプローチ

前章の課題を解決するため，本論文では，技術データ管理支援協会（MASP）の概念データモデリング（CDM）[4),5)] と，UML のクラス図・コミュニケーション図，そして，ジャクソンのプロブレムフレームの記法とを融合する．以下，この 2 つの従来技術について簡単に説明する．

3.1　概念データモデリング

技術データ管理支援協会（MASP）が提案する概念データモデリングでは，世の中のビジネスを構成する「もの」と「こと」に着目して，対象業務をモデル化する[4),5)]．具体的には，以下の図を作成する．この中で，業務内容を大局的にモデル化する際に特に重要な役割を果たすのは，3 番目までの 3 つの図である．

- 静的モデル
  業務に関係し，重要な「もの」とそれらの間の関連を表現する．**図 2** には，その要素をも模式化して示した．静的モデルは，「実体モデル（エンティティ）」と，その「関連」から構成される．エンティティは，実社会に存在する「もの」である．必ずしも，物理的に存在しているもののみではなく，「預金」のように，抽象的ではあるが，現実社会では「もの」のように識別され，状態が変化するものが含まれる．
  一方，「関連」は，「もの」と「もの」との機能的関係を示す．ただし，中村の「もの」

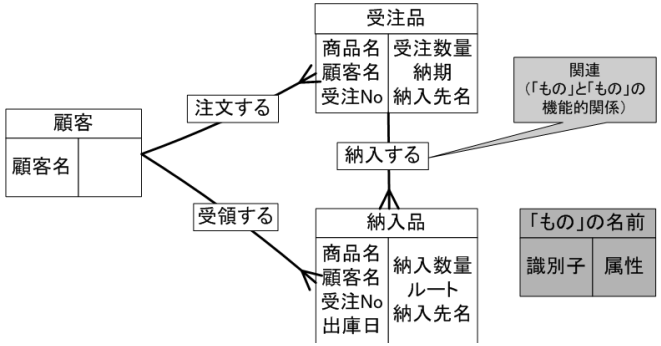


図 2　静的モデルの例（部分）
Fig. 2　The part of example of static model.

「こと」の考え方[6] にならって，「こと」とは，「望まれるデータ状態[⋆1]」である．ER 図における is-a や part-of の関係も記述する必要があることもあるが，あくまでも「業務」の姿でとらえて，現実社会にはどんな「もの」があり，そして，その「もの」と「もの」とがどのように関連しているかを記述する．

「もの」には，それを識別する名称がある．図 2 では，「顧客」「受注品」「納入品」などが，この名称である．また，それぞれの「もの」には，その「もの」を現実社会で識別するための「識別子」と，識別子以外の，その「もの」が持っている「属性」を記述する．それぞれの「もの」の左下部分にあるのが識別子であり，右下部分にあるのが属性である．

ここで，識別子は，いわゆるキー属性とそのまま対応するものではない．あくまでも，現実社会の姿を写し取るのが CDM の目的であるため，この識別子は，現実社会でその「もの」が業務においてユニークに識別するための情報を指している．いずれにせよ，この静的モデルによって，現実社会の業務に出現する「もの」が，その「もの」と「もの」との関係で写し取られることになる．なお，CDM は，情報システムの仕様を記述することを目的とするものではない．したがって，本来の CDM では，情報システムやそれを利用するユーザは記述されていない．あくまでも，業務自体の中に現れる「もの」とその関連が記述される．

⋆1 たとえば，市民税の徴収業務を考えると，市民（納税義務者）が窓口に行くような行為は「現実社会でのデータ（情報）の流れ」ではないので，「関連」になることはない．逆に，「転入届を出す」「転入する」という行為は，あるべきデータの姿を変えるので，「関連」として記述される．

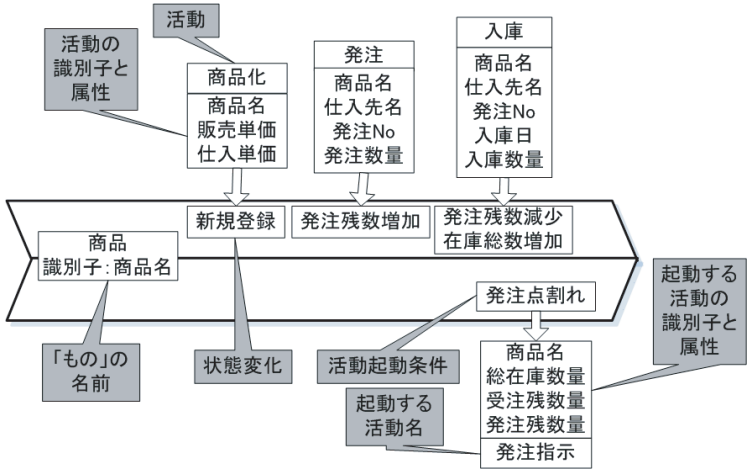


図 3　動的モデルの例（部分）
Fig. 3　The part of example of dynamic model.

- 動的モデル

静的モデルに現れた「もの」の中で，データ状態が変化する原因となる「こと」を順番に記述したモデルである．**図 3** には，模式的に，動的モデルを示す．動的モデルは，実体クラス 1 つに対して記述される．図 3 は，「商品」についての動的モデルである．基本的には，静的モデルに出現する「もの」それぞれに対して動的モデルが記述されるべきであるが，動的モデルは，「もの」のデータ状態が現実社会で変化してゆく様子を記述したものであるため，静的モデルに出現するすべての「もの」に対して記述する必要性は必ずしもない．

図 3 の中心の矢印が左から右へと流れているのは，時間の経緯を表す．上部からの矢印は，それぞれ，現実社会におけるある「活動」によって，着目している「もの」の属性値が変化していることを示している．この「活動」を現実社会で識別している識別子と，この活動によって変更されるデータが属性名称として記載されている．結果として，この「活動」によって，「もの」の属性値は変化する．「もの」は，通常は，データ変更を受けることになるが，あるデータ状態が生じたときには，ここから「活動」が起動されることがある．これが，中心に書かれた「もの」から下向きに書かれた矢印である．

動的モデルで注意が必要なのは，1) データ状態が現実社会において変化するものを写し取っているので，参照や検索といったデータ状態が変化しないものは記述されない．2)「活動」の時間関係は，左から右へ順番に記述されるが，厳密なものではなく，ましてや，一定回数繰り返したり，条件が成立した際には実行するなどの条件を記述することも考えていない．あくまで，現実社会に生じる「活動」すなわちデータ状態の変化を，特定の「もの」に対して，大まかに記述したものである．また，それぞれの「活動」には，その「活動」を現実社会でユニークに識別する識別子がつくが，これも，あくまでも，「活動」の識別子であって，静的モデルの識別子とは必ずしも一致しないことがある．

CDM では，静的モデルにおける「もの」粒度は，動的モデル（あるいは，組織間連携図）によって制御される．基本的には，動的モデルにおいて，同一の動きをするものが，1 つの「もの」となる．よく知られた「醜いアヒルの子の定理[13]」が示すように，「もの」と「もの」の類似度を，扱う側の価値感なしに一般的に区別することは本質的に不可能である．この粒度を決めるために，CDM では，ブリッジマンが与えた「操作的定義[14]」を用いているように思われる．「もの」の粒度が，その「もの」が操作されるプロセスである動的モデルによって規制されるからである．CDM で分析対象とする分野（「事業領域と使命」と CDM では呼ばれる）が変化すれば，「もの」の粒度が容易に変化しうるのである．

- 組織間連携モデル

  静的・動的モデルに現れた「もの」「こと」を実際に存在する組織上に貼り付け，データの流れや責任関係の妥当性を検証する．**図 4** には，模式的に組織間連携モデルの例を示す．「もの」が次々と変化して流れてゆくさまを記述しているので，基本的には，データフローダイアグラムに近い．しかし，以下のようにいくつかの点で，データフローダイアグラムとは異なっている．
  - CDM は，業務の本質のみを写し取り，情報システムやその操作者は含まない．このため，結果的にあるべきデータの流れを抽出することによって「as is」の情報の流れを「to be」な情報の流れに変える力を持っている．
  - 組織間連携モデルを書くことによって，逆に，静的モデルの粒度が変更されることがしばしば生じる．これは，データの流れの本質のみを取り出したときに，現実世界に存在する「もの」をどのような粒度でとらえることが望ましいかを導いている．
  - 従来の要求分析フェーズのように，現場ヒアリングしてデータフローダイアグラムを取り出すと，データフローは複雑化する．結果的に，組織間連携モデルに近いものを描いても，「to be」のデータの流れを取り出すことは難しい．逆に，この組織間連携モデルでは，シンプルな記述で全体が記述されているため，データの投入責任を複数の組織が負っているとか，データを変更する権限がない組織が存在する（そのような組織は，組織自体の存在目的が不明瞭となる）などの，データの一貫性保持の視点から，データの流れを検証可能となる．

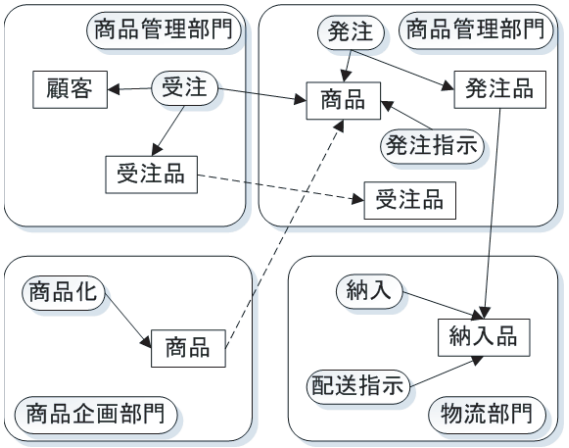


図 4　組織間連携モデルのイメージ図
Fig. 4　The example of cooperation model between organizations.

- 機能モデル

  上記以外の機能や動的モデルでは細かすぎる詳細な機能についてデータフローダイアグラムで記述する．本論文の議論とは直接には関係しないので，詳細は省略する．

3.2　プロブレムフレーム

プロブレムフレームは，ジャクソンにより提案された，問題解決デザインパターンの 1 つである[11]．デザインパターンは，通常，ソフトウェアの設計上の枠組みとして認識されているように思われる．これに対して，プロブレムフレームはコンピュータとそれを取り巻く外の世界との間を問題領域とする．すなわち，情報システムで実現するべき外の世界からの機能要求とそのために必要な情報システムの仕様要求を橋渡しする．このため，CDM が持っている抽象性を機能仕様に変換するために効果的と期待される．プロブレムフレームは

以下に示す基本の 5 つのフレームとそれらの組合せである変種フレームで構成される．

- 必要とされる振舞いフレーム
  外の世界や情報システム内の特定の条件に対して情報システムが制御される部分の問題を明確にする．
- 命令された振舞いフレーム
  外の世界のオペレータ（情報システムの操作者）が発行するコマンドに対して情報システムが制御される部分の問題を明確にする．
- 情報表示フレーム
  外の世界の状態や振舞いについて，特定の情報が継続的に必要な場合に，情報の入手方法と表示に必要な形式と適切な場所の問題を明確にする．
- 仕掛品フレーム
  コンピュータ上で処理可能なテキストやグラフィックなどをユーザが作成・編集・検索するような特定の機能にはツールが必要となる．その機能が必要となる領域の問題を明確にする．
- 変換フレーム
  コンピュータが読み書きできる入力ファイルに対し，特定の形式で変換した出力が必要となる部分の問題を明確にする．

4. 提 案 手 法

EA（Enterprise Architecture）に基づく情報システム設計では，対象ビジネスのモデルから情報システムを導く過程において，ビジネスの要求からデータ構造を導く必要がある．そこで，まず，対象ビジネス領域の「もの」「こと」を CDM で取り出し，さらに，「もの」から概念クラス図を作成し，最後に，「こと」からシーケンス図とプロブレムフレームの機能パターンを導出する手法を提案する．

提案手法は**図 5** 中の (1)〜(4) に示すような段階に分かれる．まず現場ヒアリングや業務分析を通して，MASP の CDM を実施する．この段階では業務に関する情報が中心である．次に，情報システムの実装に関する情報として「人」に着目して CDM の各モデルを詳細化する．その後，静的モデルと動的モデルを中心に情報システムの実装要素を抽出する段階に入る．まず，静的モデルの「もの」の属性や関連に注目して概念レベルのクラス図を作成する．次に，動的モデルの「こと」の順番に注目して概要のシーケンス図を作成し，機能に関する要求と仕様についてはプロブレムフレームを適用する．そして最終的に詳細なシーケンス図を完成させる．以下は各段階の概略について述べる．

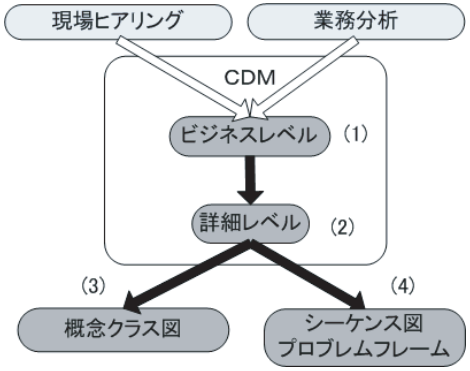


図 5　提案手法のフロー
Fig. 5　The procedure of proposed method.

(1)　MASP により提案された CDM を実施する．現場ヒアリングや業務分析で得たデータからステークホルダを導き，対象ビジネスにおける「事業領域と使命」を決定する．その中から現実社会に存在する「もの」から静的モデルを，「こと」から動的モデルを作成し，それぞれから組織間連携モデルを導く．組織間連携モデルと現実社会を照らし合わせ，「もの」「こと」の過不足に応じて静的・動的モデルを修正する．

CDM を描いた段階では，いずれのモデルも対象のビジネス・業務における情報の流れを整理・確認ができる程度の抽象度が高い状態である．したがって，この段階では，情報システムの実装についてはもとより，各モデルのどの領域が情報システムで扱われるかなどの，情報システムとビジネス・業務が行われる外部の世界との境界についても明確ではないので，情報システムの実装を意識する必要はない．さらに，CDM のモデルでとらえられるのは，情報システムでいうところのデータ更新が行われる部分について関連がある領域である．したがって，作成されるモデルはデータ更新が起こりうるビジネス・業務上の情報の流れをとらえたデータモデルとなる．

(2)　CDM を描いた段階では，ビジネス・業務の情報の流れを概念的にとらえられるだけであり，情報システムの実装要素を導くには抽象化のレベルが高すぎる．このため，情報システムの実装に向かうには，実装を意識した CDM の詳細化が必要となる．具体的には，CDM の各図において，主に以下の点に留意して追記修正し，情報システ

ムの内部で扱う情報のドメインと，外部の世界で扱う情報のドメインとの境界を決定する（CDM が表現したビジネスのすべてが情報システム化されるわけではないことに注意）．

【静的モデル】静的モデルでは，対象業務の中で「もの」を扱う「人」を必要に応じて「もの」として追加し，「もの」の属性と識別子は，実装する情報システムが管理するものに絞り込み，モデルの修正を行う．これは，初期の静的モデルではモデル中の「関連」を表現する動詞と，その主語・述語にあたる「もの」について，ビジネス・業務を行う外部の世界でどのような「人」が扱い，操作が作用する「もの」が情報システムの内部と外部どちらに存在するのか曖昧であるために必要である．明示的に「人」を「もの」として出現させることで，「人」が「もの」に行う操作によって起きる「もの」についての粒度・認識の違いを調整し，結果的に外部の世界で扱う情報と，情報システム内部で扱う情報とを分離することができる．

【動的モデル】動的モデルでは，「こと」による「もの」のデータ状態の変化や，「こと」を起こす「もの」の内部状態の変化，「こと」自身が複数の「こと」を構造として持っている場合の順序を詳細化し，「こと」の識別子と属性を絞り込む．また，見かけ上データ状態の変化はないが，明らかに業務上必要な「こと」が発見された場合，その「こと」を行う必要が生じた，など責務の発生・履行を「もの」のデータ変化としてモデルに追記する．また，「こと」についても静的モデルの詳細化と同様に，情報システムの内部で起きる「こと」なのか，外部の世界においてビジネス・業務として行われる「こと」なのか境界を明らかにする．最終的には，動的モデルを**図 6** のように並列に並べた場合に，必ず「こと」の原因となる「もの」や，「こと」が作用する先の「もの」が明確になっている必要がある．「こと」の原因と結果を明確にすることで，後のシーケンス図の雛型を作成することができる．

【組織間連携モデル】組織間連携モデルでは，通常の CDM と同じように実世界の組織に「もの」「こと」を貼り付け，組織間において「もの」の源泉の所在や「こと」によってどのように移動するかといった様子をとらえる．ただし，静的・動的モデルの詳細化で定めた情報システムと外部の世界との境界が，組織上で妥当かどうかこのモデルでチェックする．また，初期の CDM で定めた事業領域と使命に合致した情報システムを利用するビジネス・業務が可能であるかどうかもこのモデルでチェックする．

(3)　詳細化された静的モデルと動的モデルに基づき，概念クラス図を作成する．具体的には，「もの」を概念レベルのクラスに，「もの」の属性をフィールドに変換する．ただし，「もの」の持つ「こと」をメソッドへ変換する際，静的モデル上の関連が「こと」を集約した表現となっている点に注目し，関連をメソッドへ変換してもよい．なお，『対象業務の中で「もの」を扱う主な「人」』の詳細動的モデルには，見かけ上データ状態が変化しない「こと」が記述されていることがある．この種の「こと」については，関係する「もの」のメソッドとする．「もの」および関連の変換については以下の点に注意する．

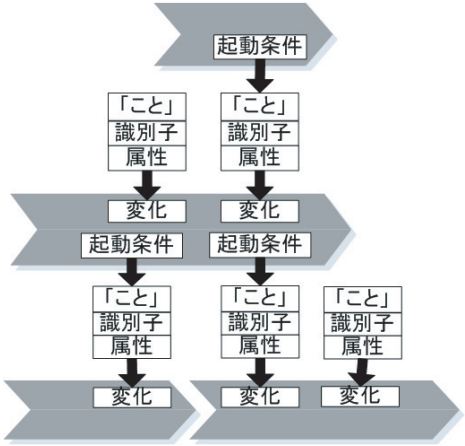


図 6　動的モデルの詳細化
Fig. 6　The refined result of dynamic model.

【1】『対象業務の中で「もの」を扱う主な「人」』と他の「もの」との間の関連の場合は，他の「もの」にメソッドを加える．対象業務の中で「もの」を扱う主な「人」を関連とともに 1 つのクラスに変換すると，システムの柔軟性を損なう結果になることがあるためである[10]．この際，クラスにとって自然な使役関係を表現するメソッド名をつける．

【2】『業務に関係する人という「もの」–対象業務の中で「もの」を扱う主な「人」』の関連の場合には，前者をクラスに置き換え，「取り込みチェックメソッド」を付与する．また間に中継データクラスを作り，「入力更新メソッド」を付与する．この関連は，データ連携に関するものなので，この他のメソッドは必要ない．このような関連では必ずデータの授受が必要で，データクラスを出現させる．

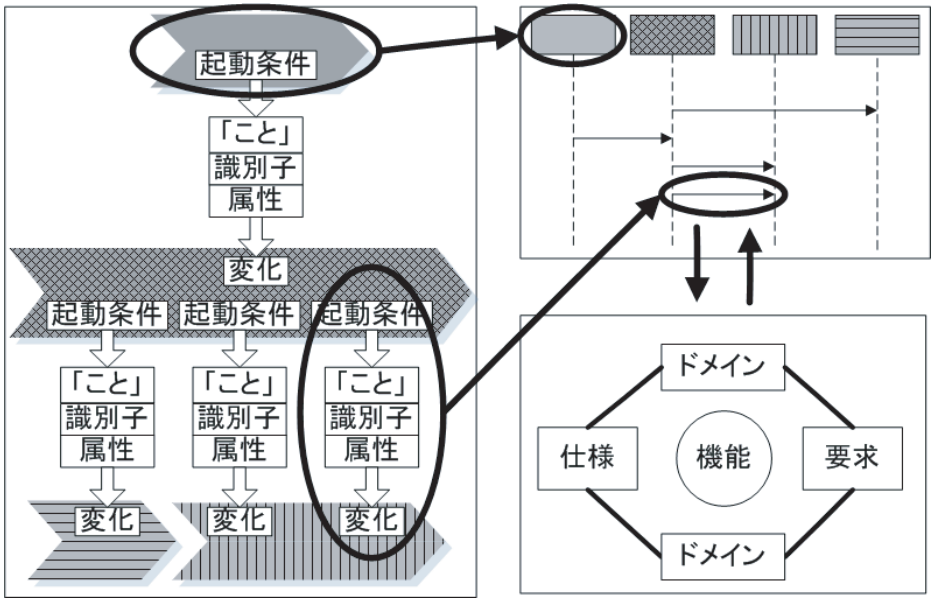


図 7　動的モデルからシーケンス図・プロブレムフレームへの変換
Fig. 7　The transformation of dynamic model to the sequence diagrams and the problem frames.

【3】人という「もの」と所有の関連を持つ「もの」は，両者をクラスへ置き換えるが，その関連をメソッドへは変換せず，リンクのみを張る．

(4)　動的モデルからのシーケンス図の作成プロセスである．**図 7** のように以下の点に留意してシーケンス図を作成する．

【1】CDM の動的モデルに表された，業務に必要な「こと」に着目し，それらの「こと」が起こりうる順序を検討したシーケンス図を，1 つの「もの」ごとに作成する．その際，順序の曖昧さが残らないよう，綿密に検討を行う．

【2】シーケンス図に現れたそれぞれの「こと」を，プロブレムフレームの 5 つの基本フレームにあてはめる．その際，必要に応じて 1 つの「こと」を複数の基本フレームに分割する．

【3】作成した基本フレームそれぞれについて，各フレームに応じたフレーム考慮事項やフレームによらない考慮事項について検討し，その「こと」を実際に行うために運用上必要となる補助的な機能を導き出す．シーケンス図から得られた順序制約についても，あわせて検討する．

考慮事項には 5 つの基本フレームに対応するもの，フレームの変種に対応するもの，そしてフレームによらず検討が必要なものがある．フレームによらず検討が必要なものには，超過考慮事項，初期化考慮事項，信頼性考慮事項，同一性考慮事項，完全性考慮事項がある．これらの項目について検討すると，業務に必要な「こと」を実際に運用していくうえで，どのような機能が必要となるかを導き出すことができる．

5.　市税税務への適用実験

上記提案手法の妥当性を検証するため，ある市の市民税に関する年次処理業務一連の業務を分析対象として，提案手法を適用した．ヒアリングにより得た市税年次処理の概要は**図 8** に示すとおりである．今回の分析における業務の事業領域と使命は，市職員の方と協議した結果，「市民税の調査・課税・収納・滞納を遂行する」となり，この目的で CDM の各図の作成を行った．

5.1　ビジネスレベル CDM から詳細レベル CDM の生成

市職員とともに CDM モデリングを行い，45 個の「もの」からなる静的モデルを作成した．「もの」は「税務課」「企業」といった組織や，「会計表」「納税証明」などの帳票・証明，「課税データ」「収納データ」といったデータが主となった．関連は「送る」「印刷する」など作業内容レベルが主であった．動的モデルでは，静的モデルの関連（作業内容）に比して詳細な「1 月 1 日住民の照合」「チェックリストの作成」などの作業手順レベルの「こと」が現れている．これをビジネスレベルでの CDM と扱った[⋆1]．

一方，業務で導入するべき情報システムを設計するには，業務上何をどのような目的で扱うかを通して，誰がどのデータをどの範囲までアクセスすることができるかを規定する必要がある．そこで，CDM の詳細化が必要である．静的モデルでは税務上の役割に応じて「税務課」に集約されていた役割と担当職員を「税務課職員」「会計課職員」など 4 つの担当課

⋆1 CDM の趣旨からすると，「もの」として組織名が入ってくるのは本来はおかしいが，税務の業務フローをとらえるため，この段階では，組織名を「もの」として採用した．後述するように，本論文で提案する手法では，最終的に，情報システムの機能を導出するため，本来は CDM には含めるべきではない業務担当者なども分析に入ってくるので，結果的には問題はない．ただし，このように「もの」の数を業務の規模の割には増加させると，本来は「to be」のデータの流れを取り出すはずの組織間連携モデルの役割を弱める危険があったものと思われる．

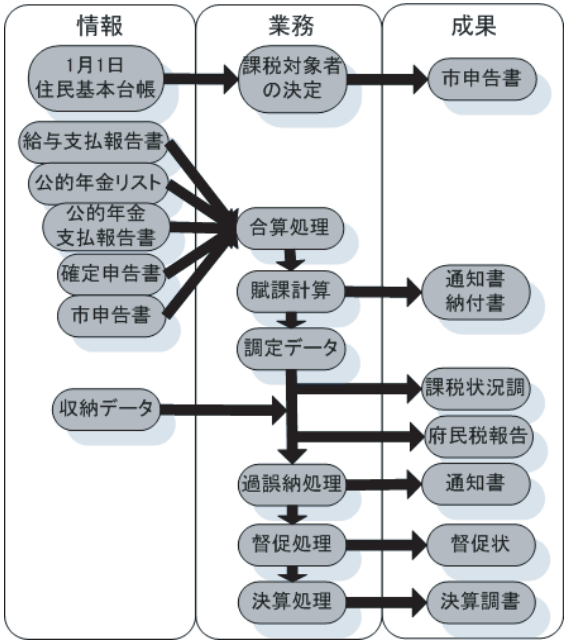


図 8　年次税務処理の流れ
Fig. 8　The procedure of tax practice of each year.

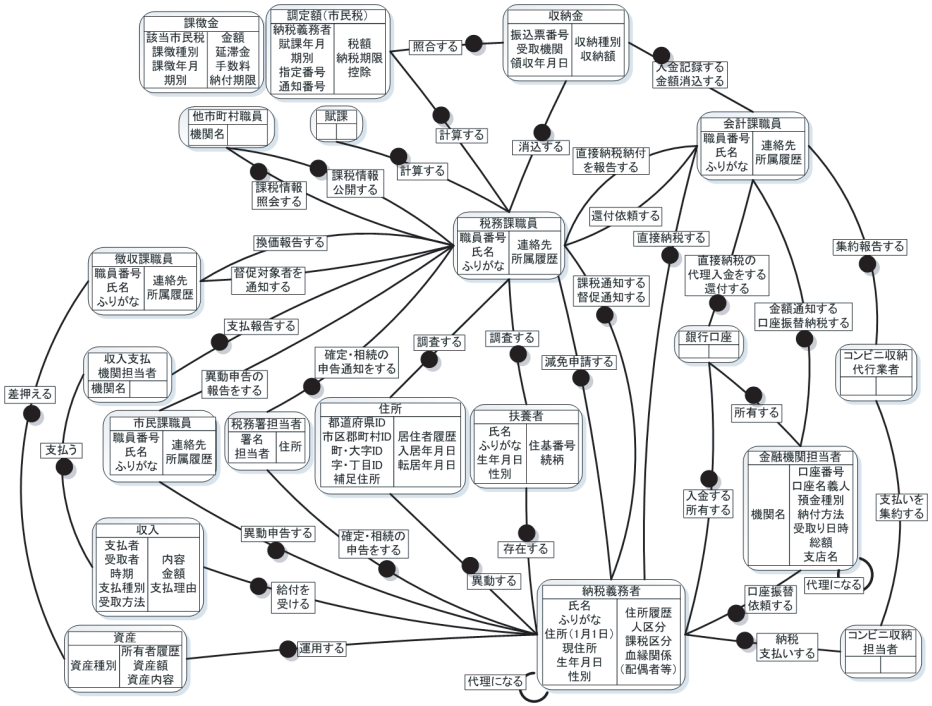


図 9　詳細レベルの静的モデル
Fig. 9　The detail of static model.

職員という「もの」に分けた．また，組織については組織名がついた「もの」を「税務署担当者」など組織内で業務に関係する担当者という「もの」に変更した[⋆1]．

静的モデルの「関連」については，「税務課職員は 1 月 1 日付けの住所を調査する」など，「もの」が主語・目的語となり，挟まれる関連は「もの」が業務上最終的に達成すべき「こと」を集約した動詞になるよう変更した．以上の変更から「もの」が 20 個となり，**図 9** の静的モデルを得た．●は関連における多重側を表す「もの」である．

次に動的モデルでは，主に「こと」の細分化を行った．細分化といっても，ビジネスレベルの動的モデル上の「こと」を実務の細かい作業に逐一分解するのではない．まず，静的モデル上の「関連」は「もの」を業務上ある目的で加工する動詞として表現されるので，その目的を実現に必要な「こと」がとらえられているかを整理した．また同時に，「こと」によって「もの」が受ける変化と，どのような「もの」の状態が「こと」を起こすかという条件を整理し，「こと」の原因や作用先・効果が明確になるように整理した．たとえば，異動申告という「こと」は，納税義務者からの異動申告を市民課職員が受け取り税務課へ報告するので，異動申告と税務課への報告は別々の「こと」に分ける．さらに税務課への報告で起こる変化は，納税義務者となる人数の増減とした．実際，扱う納税義務者の人数変化は，

⋆1 図 9 において「税務課職員」「会計課職員」「徴収課職員」「市民課職員」と，クラス図の観点からは「職員」に集約可能な「もの」が別々に存在しているように見える．しかし今回の分析では議論の末，各課の職員は同じ課税主体に属していながら，共通する業務が稀で，ほとんど異なる業務を行うことが判明したため，ブリッジマンの操作的定義に基づき，類似した静的側面を持つ「もの」でも，異なる「こと」で構成される場合は別々の「もの」とするモデル化を行った．

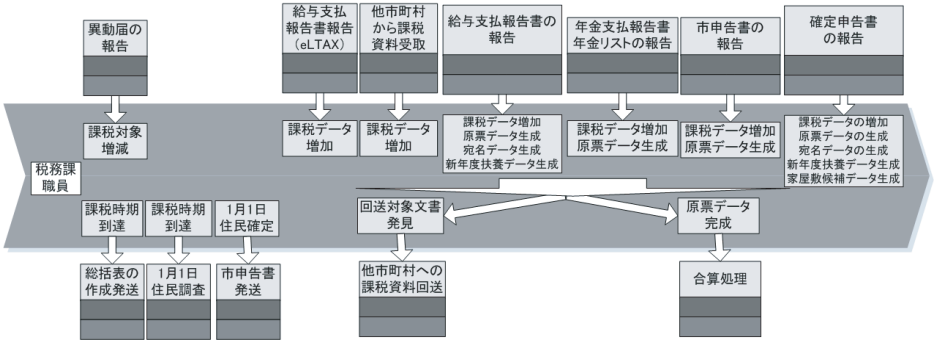


図 10　詳細レベルの動的モデル（一部抜粋）
Fig. 10　The part of detail of dynamic model.

後の合算処理を行う人数に影響を及ぼす．また特殊な例として，証明書の発行などは見かけ上データ状態に変化のない「こと」である．この場合，納税義務者が証明書を要求する「こと」を起こすと，課税主体である税務課の職員に証明書発行の責務が発生するという現象が，税務課職員の受ける変化であるととらえた．以上などの変更から，静的モデル上のすべての「もの」について**図 10** の動的モデルを得た．

組織間連携モデルでは，詳細化した「もの」「こと」を組織上に貼り付けたところ，金融機関という組織名と組織内の「もの」の名前が同じになることがあった．モデルとして組織内に組織と同じ名前の「もの」があると，組織内に存在する「もの」「こと」を曖昧にとらえている可能性があるため，金融機関の場合は内部の担当者を「もの」とする修正を静的モデルからやり直した．また，賦課は税法に則った計算による税額である調定額と，金融機関や会計課を経由する収納金に関する情報とを統合した概念である．この賦課を組織間の連携で扱う場合，調定額と収納金をとりまとめる業務の中で，計算や収納履歴から情報システムが賦課情報を生成・記録しなければならないなど，業務上の情報システムの位置づけや要求される機能についての議論と，付随した各モデルの修正を行った．

5.2　詳細レベル CDM から概念クラス図の生成

次に，詳細レベルの CDM から概念クラス図への変換を行った．基本的には静的モデルの「もの」をクラスへ変換した．たとえば，納税義務者においては，識別子に「氏名，生年月日，性別や現住所」，属性に「配偶者等の血縁関係，課税区分」を持つため，これらをクラスメンバのフィールドへ置き換えた．概念クラス図では，オブジェクトを一意に識別するためのフィールドを作る必要はないので，ここまでの記述にとどめた．ただし実際の業務上では「市民個人コード」によって個人の識別を行う手段が存在するため，実装レベルのクラス図では「市民個人コード」をフィールドとする可能性がある．しかしその場合，「市民個人コード」は概念上の制約として氏名や生年月日などと対応している必要があるため，今回の概念クラス図への変換では静的モデル上の識別子や属性をそのまま置き換えた．

メソッドの付与に関しては，「税務署担当者–納税義務者」および「収入支払機関担当者–納税義務者」の場合，中継データクラスとして「収入支払データ」と「確定申告データ」クラスを作成し，ともに「入力するメソッド」を与えた．実装を想定した場合の業務上での呼び出し順は，たとえば確定申告データの取り込みを税務課が行う場合，確定申告データクラスを生成するメソッドが呼び出され，納税義務者に対応した確定申告データを入力するメソッドを呼び出したあと，納税義務者クラスの確定申告可能メソッドで入力を許可し，確定申告クラスの入力実施メソッドの完了をもって作業が終了する．

図 11 に示した「税務署担当者–税務課職員」の「確定・相続の申告通知をする」といった関連は，特にメソッドへ変換しなかった．この関連を詳しく確認するために詳細動的モデルを確認すると，「通知およびデータの送信」という業務であることが分かる．これは納税義務者の情報を照会するために必要となる業務であり，システム上では，「納税義務者–税務署担当者」の「確定・相続の申告をする」という関連をメソッド化することで実現できる．

また**図 12** に示した「コンビニ収納代行業者–会計課職員」のような場合にも，「コンビニデータ」という中継データができる．静的モデルでは，「コンビニ収納担当者」と「コンビニ収納代行業者」を別の「もの」として扱っていたが，クラス変換の際には「コンビニ収納代行業者–コンビニ収納担当者」の関連である「支払いを集約する」は，コンビニ担当者側のシステムで行う業務であり，今回はこの 2 つを集約した．

「納税義務者–資産」については，所有という関連で結ばれているため，概念クラスとしてメソッドを付与する必要はない．ただし，資産は納税義務者の属性でもあり，リンクを切ってはならない．また資産は差し押さえ対象となるので，「徴収課担当者–資産」での関連をもとに「差し押さえ調査をする ()」というメソッドが付与される．

そのほかに，収納業務に関する「もの」として「金融機関担当者」や「銀行口座」が存在する．この 2 つは「所有」の関連であり，特別なメソッドは存在しない．しかし，これら 2 つの「もの」と「会計課職員」の間の関連は業務上重要なものが多く，変換作業は注意を要した．静的モデルでこの 2 つを分割したことによって，金融機関担当者が行う業務が明確になり，半自動化された口座経由の入金・還付作業をメソッドとして扱いやすくすることができた．

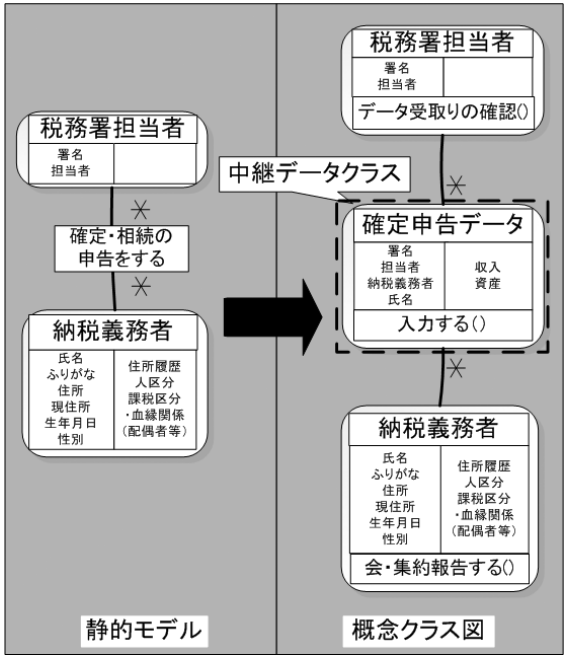


図 11　「税務署担当者–納税義務者」の変換

Fig. 11　Transformation from static model to conceptual class diagrams (between “tax officer” and “taxpayer”).

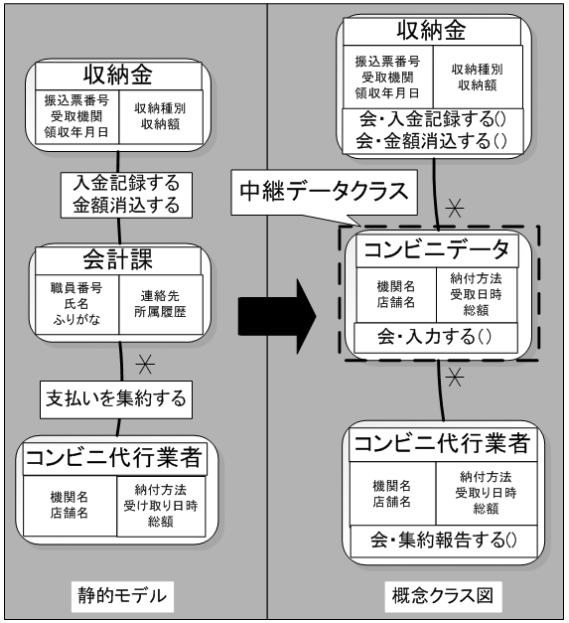


図 12　「会計課職員–コンビニ収納代行業者」の変換

Fig. 12　Transformation from static model to conceptual class diagrams (between “accountant” and “collection agency”).

本手法を用いてクラスへの変換を行う作業は，業務上のデータの流れを改めて確認し，加えてシーケンス図を利用してオブジェクトの起動，そして消滅までを大まかに考える作業でもある．作業を進めるなかで，静的モデルの識別子は必ずしもキー属性として利用することはなく，新たに実装レベルでオブジェクトの識別コードを付与する場合があることが示唆された．また業務担当職員を静的モデルへ盛り込んだ詳細 CDM によって，システムの大まかな振舞いを定義することは，概念クラス図への変換に大きく寄与したと考えられる．

5.3　詳細レベル CDM からのシーケンス図・プロブレムフレーム生成

次に動的モデルから，シーケンス図などを導く必要がある．詳細レベルの CDM からシーケンス図・プロブレムフレームへの変換は，まず動的モデルに着目し，「こと」が実行されうる順序が明確になるよう注意しながら，シーケンス図を作成する．実際に作成した図の例が以下の**図 13** である．

図 13 からは，コンビニ入金速報データが送られてくるたびにコンビニ仮入金データを作成し，最後にそれらをまとめたコンビニ入金確報データを受け取って最終的なコンビニ入金データを作成するという業務の流れが分かる．この図をもとに検討を行うことで，たとえば，コンビニ入金データの作成には，コンビニ入金確報データに対応するコンビニ仮入金データが存在している必要があることなどが導き出される．

このように，シーケンス図に「こと」の順序関係を明確に表すことで，それぞれの「こと」がどのような前提条件を満たしていなければならないか，また順序関係があやふやな箇所はないかを確認することができる．

次に，シーケンス図に表されたそれぞれの「こと」を，プロブレムフレームにあてはめる．フレーム図は**図 14** のように表される．たいていの場合，1 つの「こと」の実行のため

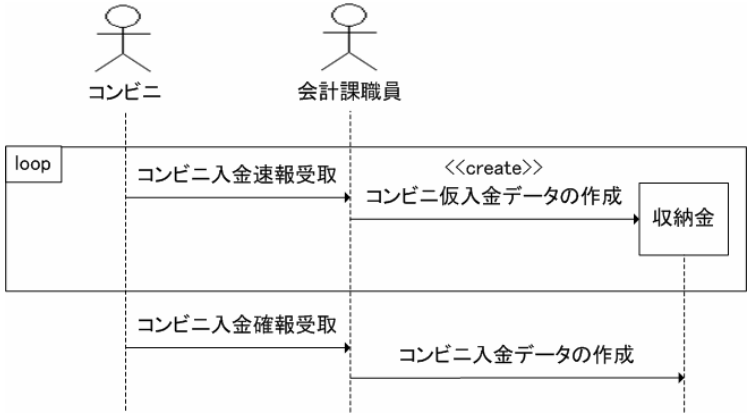


図 13　収納業務のシーケンス図（一部）
Fig. 13　The part of sequence diagrams about receive practice.

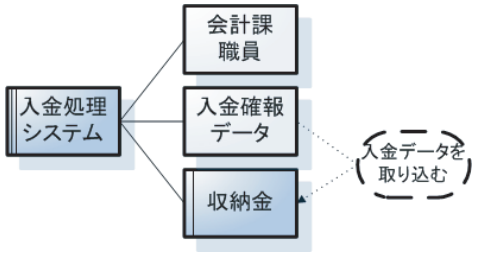


図 14　コンビニ入金確報データの取り込みのフレーム図
Fig. 14　The frame diagram about the acquisition of the data of tax payment in the collection agency.

には，インタフェース表示部，データ変換処理部といったように，複数の基本フレームへの分割が必要となる．「こと」をフレームにあてはめる際には，これ以上問題を分割する必要がないか十分に注意する．

最後に，それぞれのフレームに対し，プロブレムフレームの考慮事項をもとに運用上必要となる機能を検討していく．図 14 のコンビニ入金確報データの取り込みについて実際に検討したところ，その一部として次のような事項について考える必要があることが分かった．

- 入金確報データにデータ形式を満たさないデータが混ざっている可能性はあるか．
- 入金確報データに誤りがあった場合，どのように対処すべきか．

またシーケンス図から得られた前提条件として，入金確報データと作成済みの仮入金データの対応がとれている必要があることが分かっている．この前提条件を満たすため，以下についても検討が必要である．

- コンビニ入金確報データの個々のデータに対応する仮入金データがない場合はあるか．また逆に，仮入金データに対応する確報データがない場合はあるか．
- 上記の対応関係を満たさなかった場合，支払い情報の消失などを防ぐためどのような対応を行う必要があるか．

以上のように税務の収納業務に対し検討を行った結果，シーケンス図とプロブレムフレーム収納業務の「こと」を実際に運用する際に必要となる機能を体系的に導出できることが確認できた．

6. 考　　察

前章の適用事例で，今回の提案手法では，税務処理業務における「もの」や「こと」から，業務上必要となる情報システム内のクラスや機能種別，付随データをある程度得られることが分かった．

CDM の各図を詳細レベルに変換する部分では，たとえば税務上では税務課職員が 1 月 1 日の調査を必ず行い，課税条件として重要な要素として扱われるため，住所などは「もの」として静的モデル上に現れた．業務に関する制度や制約と，業務上必要な「もの」と「こと」の関連から分析することで，データが扱われる意図や業務上の役割を考慮しやすくするための設計が行えたと考えられる．

詳細レベルの CDM から概念クラス図の変換では，主に静的な側面について，クラスに必要なメンバやメソッドが，「もの」を構成する識別子や属性や，「こと」「関連」の動詞が表現する業務上必要とされる振舞いからとらえられている．静的モデルを主にした詳細化や変換は，クラス図だけでなく ER 図も可能であるように見える．静的モデルは「もの」と「関連」，ER 図は「実体」と「関連」を主眼とするためである．しかし，「関連」で表現される領域は双方で異なる．静的モデルの「関連」は「もの」を変化させる操作を動詞で表現しているのに対し，ER 図の「関連」は実体のデータ構造とデータどうしの論理的な関連を表現している．動詞で表現される情報はデータの論理的構造だけでなく，動詞に係る情報の振舞いをも含むため，静的モデルからの変換は ER 図ではなくクラス図へ向かうのが適していると考えられる．

詳細レベルの CDM からシーケンス図・プロブレムフレームの変換では，主に動的な側面について，業務上必要な「こと」を実現する機能が，プロブレムフレームによるパターンから導かれている．「こと」によって表現される動的な要素の考察では，最終的に必要な機能の前後関係まで導かなければならない．そのためには，機能に関わる対象の明確化と動的に関係するタイミングの把握が不可欠となるため，動的モデルにおける「こと」とその源泉・作用先の明確化が有効であると考えられる．加えて，機能を必要とする領域が情報システム内部のみなのか，ユーザインタフェースなど外部との接点を含むのかといった違いによって考察すべき要求と機能間の事項は異なり，プロブレムフレームのパターンを利用することで，漏れのない要求を実現する機能の動的要素考察が可能であると考えられる．

詳細レベルの CDM 以降のステップに共通するのは，情報システムの要素として抽出できるのが，ビジネス・業務上の要素のうち「もの」「こと」というデータ更新の起こる要素を中心にしている点である．これは同時に，情報システム内においてクラスや機能をコンピュータ上で厳密に制御するためのデータは直接的にはとらえきれないことを意味する．例としては情報システムに必要な要素のうち，検索・照会・履歴の機能の具体的な要素などが相当する．たとえば検索の場合，事前に検索する「こと」の結果を利用している「こと」を特定できていなければ必要な「こと」として導出することができない．このような要素の導出には，提案手法における動的モデルの詳細化で「こと」を細分化する際に，「こと」を実現するための手段として検索などが必要でないか，といった考慮パターンを持ち込む必要がある．最終的な実装アーキテクチャの導出には，たとえばクラス図の場合，提案手法の結果と MVC フレームワークなどのデザインパターンから実装アーキテクチャへの変換を考慮することで，情報システム制御上のデータを補って実装クラス図へ変換可能であると考えられる．

CDM で得たモデルから情報システムの実装要素の導出を目指す今回の提案手法と，従来の要求分析と実装設計を独立に行う手法とでは，提案手法の方がビジネス・業務上の問題や制約といった文脈を実装設計に活かせる点で優位であると考えられる．例として，実装設計を独立に行う手法をユースケース駆動とし，ともに市民税の賦課・収納を扱う情報システムの動的側面としてシーケンス図に必要な要素の導出を行う場合について**図 15**，**図 16** を用いて考える．ただし，図 15，図 16 のユースケースや動的モデル・プロブレムフレームは全体から必要最小限の部分を抜粋しているが，設計対象は同じである．ある時点で，滞納している納税義務者を通知・表示させるような機能の要求が出たとする．この場合，滞納者を納税義務者のどのデータがどのように変化した状態で通知・表示すればよいかを考える必要がある．単純に納税期限を超過した納税義務者を特定するだけならばそれほど複雑ではな

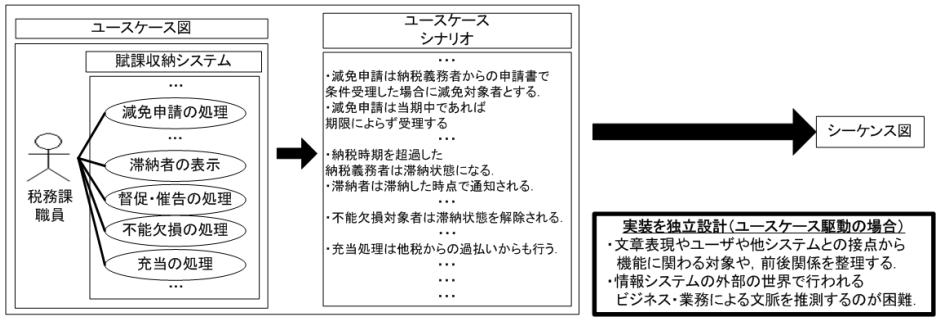


図 15　実装設計を独立に行って動的側面を考える場合（市民税一部抜粋例）
Fig. 15　The design result of dynamic aspect based on the conventional method.

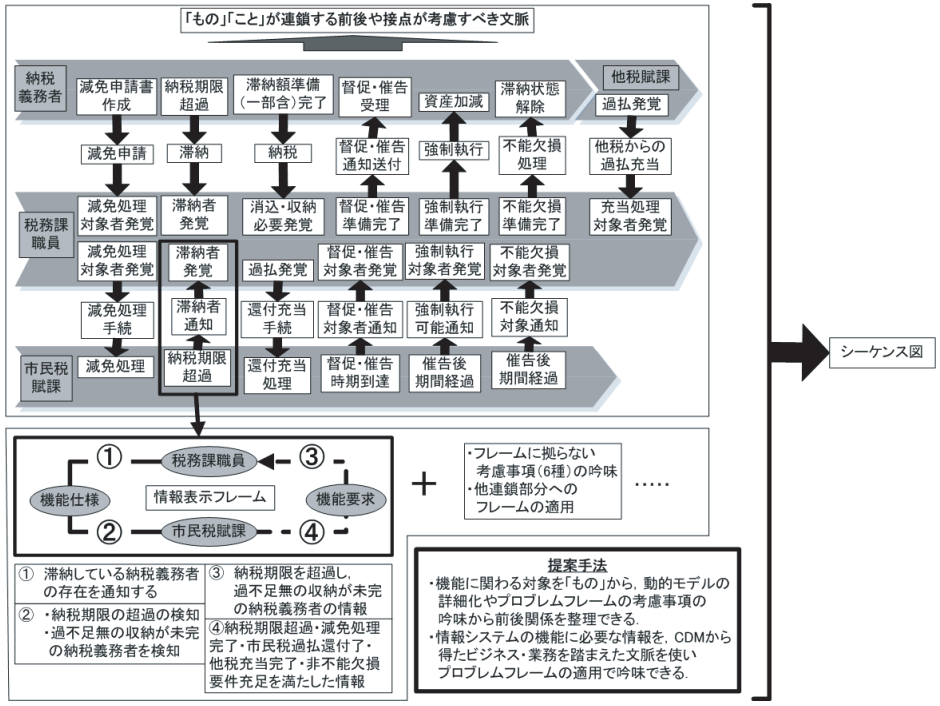


図 16　提案手法で実装設計の動的側面を考える場合（市民税一部抜粋例）
Fig. 16　The design result of dynamic aspect based on the proposed method.

い．しかし実際には，時間にかかわらず支払うべき税額について減免されている可能性や，納税を忘れていたとして遅れて収納する可能性，他税は納税していて，なおかつ過払い分が充当されている可能性などがあり，これらの非同期に発生している事項を正しく反映したうえで滞納者の通知・表示をしなければならない．非同期に起きる事項を考慮してシーケンス図を目指す場合，図 15 のようにユースケース駆動の場合は文章表現から組み立てなければならず，また情報システムとユーザの接点よりも外部で起きる事象をとらえることは難しい．一方図 16 のように CDM から詳細化・プロブレムフレームを適用した場合，あらかじめ税務処理業務の文脈として減免処理や他税からの充当が文脈としてとらえられており，それらを考慮してプロブレムフレームごとに必要な情報を用意されたパターン・考慮事項によって吟味しながらシーケンス図を作成することができる．さらにプロブレムフレームでは作成した複数のフレーム図の統合や，あるフレーム図を複数のフレーム図に分解して詳細な情報の吟味を行う技法を内部に持っているため，動的モデルでとらえている「こと」が正しくビジネスや業務を反映しているか遡って吟味することも可能である．以上が，提案手法の方が動的な側面でビジネス・業務上の文脈を実装設計に活かしやすい理由だと考えられる．ただし，CDM から詳細化する場合にはプロブレムフレームで考慮すべき事項の数は動的モデルの「こと」の数によって大きく変化する．したがって元々の CDM のモデルにおいて，たとえば静的モデルが 100 以上の「もの」から構成されるような場合ではプロブレムフレームで考慮すべき事項が数千から数万にも至る可能性がある．したがって 5 章で行った適用事例からも，人が俯瞰的に認識できる限界として，静的モデルのノードは多くても 50〜60 になるようにするのが望ましい．以上から，提案手法の適用は特に中規模程度までの情報システムの必要が想定される環境が最も望ましいと考えられる．

7. 関連手法との比較

要求分析手法としては，ラショナル統一ソフトウェア開発プロセス[15] がよく知られている．ラショナルのアプローチでは，「ユースケース駆動」によって，ユースケースから要求分析を開始し，UML による記述を何度も見直す形で要求を明確化してゆく．これに対して，MASP のアプローチでは，最初は，情報システム（機械）を意識することなく，「対象とする問題そのものに注目する」（ジャクソン）．

また，よく知られたシステムの開発方法論にジャクソンの Jackson System Development（以下 JSD）法[16] がある．JSD ではシステムの仕様化と実現は別であるとし，6 段階ある開発手順のうち，5 段階は要求されたシステムの仕様化に割き，残りの 1 段階でその仕様を実現する．これはシステムを構築する際の関心がシステムの利用者側にある人とコンピュータ側にある人とで異なる場合があり，そのギャップの解消が必要であるという立場からきている．システムとその外部の実世界は明確に区別され，システムから切り離した実世界をとらえたモデルがシステムで構築すべき主題を与える．つまり，JSD の関心は実世界がありのままにシステムに反映されるかどうかということである．ここでの実世界は与えられ固定された出発点であると見なす．ただし，実世界の一部や全部が新しく生み出され変更されることを否定するわけではない．実際，システムを必要とする実世界をとらえたうえでその内部の創造や変更を具体化する方法を JSD は持っている．同様に，JSD はシステム開発に関連するすべての作業を包含しているわけではなく，コスト計算や運用管理，ユーザインタフェースの対話設計といった側面には言及しない．あくまで実世界をありのままにとらえたモデルからシステムの機能を導くアプローチである．

JSD ではシステムの仕様化について，**図 17** のような実体行動段階，実体構造段階，初期モデル段階，機能段階，システムタイミング段階の 5 つの段階がある．特徴は機能に関する考察を後に回し，実世界をモデルとしてとらえる考察を行う点である．これは機能と比して実世界のモデルが安定しているという立場からきている．

JSD で実世界をモデルとしてとらえるには実体行動段階，実体構造段階，初期モデル段階の 3 つの段階が必要であるとしている．実体行動段階では，実世界の中で行動を起こす実体をとらえ，それが実世界の中でどのような意味を持つのか定義を明確にし，一覧表にす

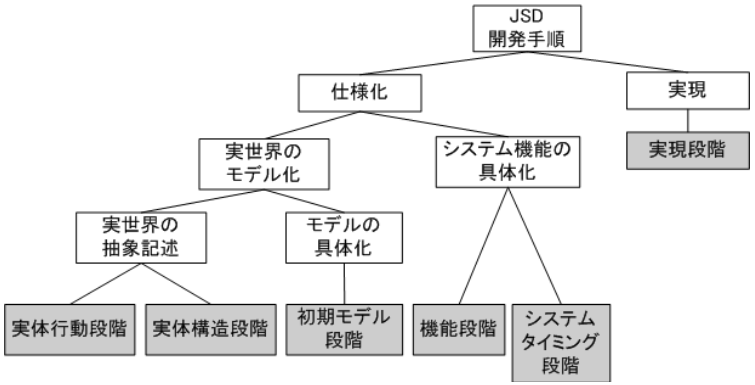


図 17　JSD 法の開発プロセス
Fig. 17　The procedure of JSD.

る．主にシステムにおける静的な側面をとらえるのが目的である．実体の定義は以降の段階で機能などを考察する基礎となるため，どのような事物を実体ととらえるかは注意が必要となる．実体構造段階では，実体行動段階でとらえた実体が起こす行動を連接・選択・反復のいずれかの構造によって，行動がどのような時系列で起きるのかを整理する．主にシステムにおける動的な側面をとらえるのが目的である．実体が起こす行動は構造を持ち，ある逐次プロセスの集合で表現されるため，システムに必要な機能に関するおおむねの時系列制約をとらえることができる．初期モデル段階では前の 2 つの段階を受け，ある実体の行動が別のどの実体に影響を及ぼすかをとらえ，つなぐことで実世界のモデルを作成する．ここで作成するモデルは行動と実体の関係をデータ列やベクトル結合としてとらえるため，一種の DFD（データフローダイヤグラム）のような形式をとる．ただし，より具体的なシステムの機能は後の段階で考察される．

JSD と CDM には類似点が多く存在する．実体行動段階の主眼は実体をとらえることであり，静的モデルでとらえる「もの」に類似している．実体構造段階の主眼は実体が起こす行動の具体化であり，動的モデルでとらえる「こと」に類似している．初期モデル段階で作成するモデルはおおむねではあるが DFD の形式をとり，組織間連携モデルで作成する図と酷似している．JSD と CDM で異なるのは，JSD は実世界が与えられたときのシステムの実装までの手法を与えるのに対して，CDM はとらえるべき実世界のドメインを考察するところから始まり，実世界の組織上で扱う実体や行動を含む「もの」「こと」をシミュレートする手法までしか与えない点である．この差異が明確に現れているのは JSD における実体と CDM の静的モデルにおける「もの」がとらえる対象範囲である．どちらも「物」だけでなく概念を含む事象をとらえることができる．しかし，JSD では実体は行動の要素がないものは実体と見なさない．すなわち状態しか持たないものは実体にならない．一方 CDM の「もの」は一見状態しか持たないものであっても，それが他の「もの」からの行動によって変化するのであれば静的モデルの中に含めることができる．この実体と「もの」のとらえ方の差異は実世界が与えられたものとする JSD と実世界の目的や使命などから考察する CDM とでは当然発生するものである．しかし JSD では後の実体構造段階などで，時系列は連接・選択・反復のいずれかで構造化できるとしている．これは情報システムありきの分析から実装では効果が期待できる．しかし，CDM のようなビジネスのアーキテクチャからとらえる手法では，情報システム化する範囲が複数存在するか，極端には情報システム化する範囲が存在しない可能性すらありうる．したがって，時系列の構造化などが困難な場合も存在するため，JSD が効果を発揮する場面と CDM が効果を発揮する場面は異なるといえる．

8. おわりに

今回の提案手法では，税務処理業務のうち，業務を行う際に扱う「もの」が更新されるようなデータについて，実装すべき情報システムの概念クラスと機能種別については一定の情報を抽出することができた．しかし，完全な実装レベルの設計には，今回の結果で得たデータに加え，業務レベルで必要な「もの」「こと」以外の，情報システム内でデータを制御するための情報まで導かなければならない．

そのためには，詳細レベルの CDM で表現される業務上の「こと」の達成に必要な「こと」を導出するようなプロセスや，今回の結果に加え，MVC モデルのような実装アーキテクチャ上で考えた場合の概念クラス図やシーケンス図プロブレムフレームを通した「もの」「こと」の位置付けを行う手法を考える必要がある．

参考文献

1) 経営情報学会・システム統合特設研究部会（編）：成功に導くシステム統合の論点，日科技連合 (2005).
2) 前掲書，p.121，KDDI の事例—概念データモデルによるシステム統合.
3) 特定非営利法人技術データ管理支援協会（MASP），Web サイトは以下のとおり.
http://www.masp-assoc.org/
4) 手島歩三：概念データモデル設計によるソフトウェアのダウンサイジング，日本能率協会マネジメントセンター (1994).
5) 手島歩三：ビジネス情報システム工学概説—概念データモデリングに基づく情報システム構築と運営，技術データ管理支援協会（MASP）内部資料（非売品）(2006).
6) 中村善太郎：もの・こと分析で成功するシンプルな仕事の構想法，日刊工業新聞社 (2003).
7) 杉原　明，白崎俊行，森　弘之：J-Smile を支える IT イノベーション（メソドロジ)—柔軟なシステム構造，短工期開発を実現する設計開発方法，JFE 技報，No.14, pp.25–28 (Nov. 2006).
8) H・ウィリアム・デトマー（著），内山春幸，中井洋子（訳）：ゴールドラット博士の論理思考プロセス，同友館 (2006).
9) 吉澤憲治，星　翔太，金田重郎：TOC と CDM を用いた業務分析手法の提案，情報システムと社会環境研究会 (2008).
10) 児玉公信：UML モデリングの本質—良いモデルを作るための知識と実践，日経 BP 社 (2004).
11) マイケル・ジャクソン（著），榊原　彰，牧野祐子（訳）：プロブレムフレーム—ソフトウェア開発問題の分析と構造化，翔泳社 (2006).

12) マイケル・ジャクソン（著），玉井哲雄，酒匂　寛（訳）：ソフトウェア要求と仕様—実践，原理，偏見の辞典，新紀元社 (2004).
13) 渡辺　慧：認識とパタン，岩波新書 (1978).
14) S.I. ハヤカワ：思考と行動における言語，原著第 4 版，岩波書店 (1985)，第 10 章，「われわれはどうやって知るか」.
15) イヴァー・ヤコブソン，グラディ・ブーチ，ジェームス・ランボー（著），藤井　拓（監訳）：UML による統一ソフトウェア開発プロセス—オブジェクト指向開発方法論，翔泳社 (2000).
16) マイケル・ジャクソン（著），大野徇郎，山崎利治（監訳）：システム開発—JSD 法，共立出版 (1989).

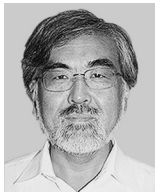
(平成 20 年 5 月 20 日受付)
(平成 20 年 11 月 5 日採録)

吉田　和正

2007 年同志社大学工学部知識工学科卒業．同年同大学院工学研究科博士課程前期課程進学．現在，在学中．業務用 Web アプリケーションの開発や，情報システム開発方法論等の研究に従事．

金田　重郎（正会員）

1976 年京都大学工学部電気第二学科卒業．1978 年同大学院工学研究科博士課程前期課程修了．同年日本電信電話公社武蔵野電気通信研究所入所．1998 年同志社大学大学院総合政策科学研究科・同工学部．工学博士（京都大学），技術士（情報処理）．情報システム開発方法論，データマイニングとその応用，ユビキタスサービス等の研究に従事．



芳賀　博英（正会員）

1978 年同志社大学工学部電子工学科卒業．1980 年同大学院工学研究科博士課程前期課程修了．同年（株）日立製作所入社．1994 年同志社大学工学部．工学博士（京都大学），U.K. Chartered IT Professional（英連邦情報技術士）．E-Learning，データマイニングとその応用，ユビキタスシステム，MOT 等の研究に従事．