概念データモデリングによる抽象クラス設計手法の検討

上田　紳一朗†　吉田　和正‡　吉澤　憲治‡　金田　重郎‡
同志社大学工学部†　同志社大学大学院工学研究科‡

1. はじめに

情報システム開発分野では，従来，現場ヒアリング中心の開発手法を良しとしてきたが，業務全体のデータ不整合や改修時間の肥大化などの問題があった．この解決策として，特定非営利法人技術データ管理支援協会（以下，MASP）の概念データモデリング（以下，CDM）がある[1]．CDM はビジネスの視点から本質的に重要な「もの」，その振る舞いとしての「こと」を捉え，ビジネス対象を可視化するトップダウンの設計手法であるが，実装については未検討である．そこで本稿は CDM における静的・動的モデルと，UML におけるクラス図，コラボレーション図の相関から概念レベルのクラス図を設計する手法を，ある市の市民税業務を例に検討する．

2. 情報システム開発の現状

情報システム開発では近年，業務プロセスよりも，データ基盤を第一に設計するのが一般的である[2]．多くのシステム開発において，使用している帳票などからデータ基盤を設計し，更に業務フローの分析やヒアリングを通して必要な機能を設計していく．しかし，現実には業務は日々変化し，システムへの機能変更要求によって，データの整合性は失われていく．開発者のセンスに左右されず，そして業務内容の変更に強いシステム設計手法が必要とされている．

MASP の CDM はこのような要求に答える目的（トップダウン）思考のアプローチである．しかし，CDM は実装については未検討である．そこで本稿では，ある市の市民税業務を対象に CDM を行い，さらに実装へ向けて手法を検討し，システム設計をどれだけ支援できるか評価する．

3. 概念データモデリング（CDM）

MASP が提案する概念データモデリングは，実社会に存在する情報を，ありのままに写し取るための手法である．具体的には次の図を作成する．特に重要な役割を果たすのは，上から3番目までの3図である．

- **静的モデル**：業務に関係した「もの」を識別子・属性と「もの」同士の関連で表現する．
- **動的モデル**：静的モデルに書かれた「もの」が状態変化していく様子を，原因となる「こと」を元に記述する．
- **組織間連携モデル**：上記の動的モデルを実際に存在する組織の上に貼り付けて，データの流れ等の妥当性を検証するもの．
- **機能モデル**：上記以外の機能についてデータフローダイアグラムで記述したもの．

設計当初に重要なのは，概念レベルでの変更の少ない「もの」と「こと」を捉えることであり，CDM はそれを可能にする．また図 1 のように，CDM の視点は UML のそれよりさらに上流である．業務の本質を捉え，開発側だけではなくシステム発注側にも理解可能なモデルとなっている．

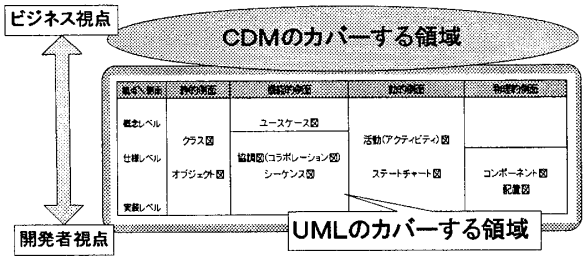

図 1：CDM と UML の表現対象レベルの違い

4. 提案手法

図 1 のように，CDM はビジネス視点のモデルである．設計を考える際には，システムがどのように振舞うか，つまりユースケースを記述する必要がある．本手法では意図的に静的モデルに対して業務担当者のエンティティ（以下，実体）を盛り込み，「業務担当者」と他の実体との関連を記述することとする．また動的モデルの「こと」がコラボレーション図のメッセージと対応する点を踏まえ，業務の本質である「もの」を概念クラス化し，変更に強いシステムを構築する．しかし，例えば領収書の発行などのようなデータ更新が起らない「こと」は CDM では記述し得ない．そこでこのような業務を捉えた動的モデルを新たに作成する．

Deliberation of the approach to design abstract class by using CDM
Shinichiro Ueda*1 Kazumasa Yoshida*2 Kenji Yoshizawa*2 Shigeo kaneda*2
*1Faculty of Engineering, Doshisha University
*2Graduate School of Engineering, Doshisha University

以上の考察から，次のような概念クラス設計手法を提案する．

【STEP1】対象業務に対して，CDM を行う．業務担当者を実体として扱わないよう注意する．
【STEP2】組織間連携モデルから理想的な業務担当者の業務モデル（こと）を策定する．
【STEP3】静的モデルの実体をクラスへ，属性をフィールドに変換する．この段階ではまだ『こと』をメソッドに変換しない．
【STEP4】業務担当者を盛り込んだ実装寄りの静的・動的モデルを作成する．尚，業務担当者の動的モデルを書く際の手順は次の通りである．
　1）「データ更新を伴わない『こと』」を追加する．
　2）上記の『こと』で，読み出される属性を記述する．
　3）上記の動的モデルに追加された識別子・属性はすべて，静的モデルに追加する．
　4）上記の静的モデルをクラス図にするため，実装上に必要なキー属性などを追加する．
【STEP5】「業務担当者-他の実体」間の各関連を参考に，「他の実体」にメソッドを加える．この際，実体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．またフィールドについても追加があれば，修正する．

図 2：概念クラスへの変換の流れ

業務担当者の動的モデルは実装に近いワークフローとなる．また【STEP5】以降，概念クラス図から抽象クラス図への変換は，一定の規則に従って変換可能であり[3]，本手法には盛り込んでいない．

5. 市民税の処理業務への適用

適用対象は，ある市の税務処理業務であり，特に市民税業務をモデリング対象とした．本手法から実際に作成した概念レベルのクラス図を図 3 に示す．

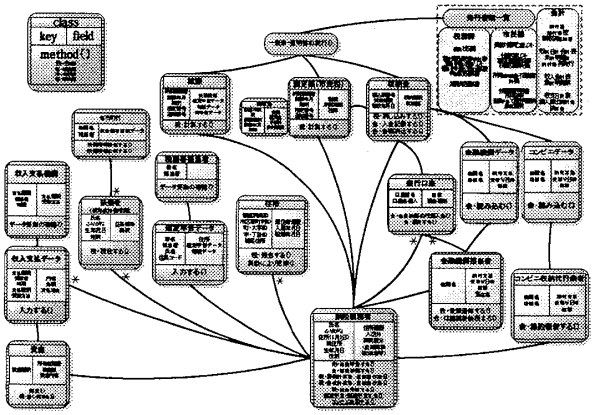

図 3：市民税業務への適用結果

【STEP1】において，静的モデルは 15 個の実体とそれらの関連から構成された．また．【STEP5】では，【STEP4】で新たに追加された 11 個の関連が各実体に付与された．

この結果に対する比較対象として，業務フローから作成した MVC モデルを利用する．MVC モデルとは，システムの中でビジネスロジックを担当するモデル(Model)，表示や入出力を担当するビュー(View)，これらを制御するコントローラ(Controller)から成り立つ開発デザインパタンである．本手法との比較の結果，次のように集約された．

- Model について，実体はほぼ一致したが，履歴や処理端末番号，日時などのシステム上必要なフィールドが本手法では欠落していた．
- View において，通知書や一覧表の発行画面を本手法は抽出することができた．しかし，履歴，参照，検索画面については抽出に失敗している．
- Controller では，業務レベルで必要な「メソッド」には対応することができた．しかし，これまで同様，システムレベルで必要なメソッドを抽出することはできなかった．

MVC モデルは，実装に近いシステムのモデルであり，システム側から業務を捉えているとも言える．これに対して CDM は，業務側から見たシステムのモデルである．そのため，本手法はシステムレベルで必要なフィールドやメソッドを抽出し得ない．

本手法にこれらの情報を付与するステップを加え，フロー化することは可能である．しかし，単純に全ての実体に対して履歴などのフィールドを追加することは処理速度の観点から適当ではない．なんらかの規則を新たに考える必要がある．

6. まとめ

本手法により，CDM から抽象クラスへの変換は可能である．その結果，業務レベルから見たフィールドやメソッドを自動的に抽出する．しかし，システム上必要な履歴などに関するフィールドやメソッドを追加する人為的な作業が必要となる．

参考文献

[1] 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
[2] 林衛，“改訂ＥＲモデルによるデータベース設計技法”，ソフト・リサーチ・センター，2005
[3] 児玉公信，“UML モデリングの本質”，日経 BP 出版センター，2004