情報処理学会論文誌 Vol.56 No.5 1351–1362 (May 2015)

ビジネス系アプリケーション標準構造開発への取組み

亀井 邦裕[1,a)] 児玉 公信[2] 細澤 あゆみ[2] 成田 雅彦[1]

受付日 2014年8月18日, 採録日 2015年2月4日

概要：企業情報システムは変化する企業環境に対し，柔軟で機敏な対応を迫られている．そこで稼働するビジネス系アプリケーションは短期間での構築が可能で，なおかつ変化に強い構造を持っており，再利用開発が可能でなければならない．それは経験に裏打ちされた合理的な概念構造を持ち，変化する部分と固定部分が明確に分かれた実装構造となっているはずである．本論文は，そのようなアプリケーションを開発するための１つの取組みとして，概念モデルに基づく実装方法を試行し，消費税計算などの公開可能な題材を用いた概念モデル，実装モデルなどを成果として提示する．

キーワード：企業情報システム，ビジネス系アプリケーション，再利用開発，概念モデル，実装モデル

An Approach to The Standardized Structure for Business Application Software

Kunihiro Kamei[1,a)] Kiminobu Kodama[2] Ayumi Hosozawa[2] Masahiko Narita[1]

Received: August 18, 2014, Accepted: February 4, 2015

Abstract: Enterprise information system must be flexible and agile in order to address the environmental change. Therefore business application software in the system should be rapidly developable, changeable in structure and reusable. It must have a experimentally derived, reasonable structure that may be separated into changeable and unchangeable parts. In this paper, we try the implement method based on conceptual models, by which the reusable applications can be developed, and as the result we provide examples of conceptual models, implement models, and other deliverables, using subjects that can be disclosed such as the consumption tax.

Keywords: enterprise information system, business application, reusable, conceptual model, implement model

1. はじめに

現在，企業情報システムをとりまく環境（市場の要求，法制度など）は多様化し，ますます変化の速度を速めている．成長戦略による規制の緩和などで企業環境は大きく変化する．ビジネスの基本的活動は変わっていないとしても，たとえば，その取扱商品は，サービスとの組合せにより構造が複雑化し，取引形態も多様化している．かつて固定電話のみを扱っていた通信企業は，今やモバイル電話，IP 電話，インターネットプロバイダ，CATV などを融合させたサービスを，様々な割引をともなう料金体系で提供しなくてはならない状況にある．

一方，現状このようなビジネスの変化に絶えずさらされている企業情報システムの一部であるビジネス系アプリケーションソフトウェア（以下，単にアプリケーションという）は，短期間に構築でき，かつ変化に柔軟に対応できる構造を持たなければならない．それは，ある範囲で業種や企業を越えて多くのビジネスに対応できる標準化された構造をしていて，それにより再利用が可能であり，かつ変化に強い構造になっているものである．

現状のアプリケーションはどうだろうか．記述言語の主流は COBOL，C などの手続き型言語から C++，Java，

[1] 産業技術大学院大学
Advanced Institute of Industrial Technology, Shinagawa, Tokyo 140–0011, Japan
[2] 株式会社情報システム総研
Information Systems Institute, Ltd., Shinjuku, Tokyo 160–0023, Japan
[a)] kunihirokameipmp@yahoo.co.jp

C# などのオブジェクト指向プログラミング技術を実装した言語に移行した．しかし，実態は各企業の基幹業務を担うアプリケーションには，以前の手続き型言語で書かれたものが膨大に残存している．それらの手続き型言語のソフトウェアやその後主流となった JAVA 言語のソフトウェアの多くは，大規模・短納期に対応するためという理由で，正しく機能別のモジュール設計が行われないまま，ウォータフォール型プロセスで開発された．それらは入出力画面ごとに対応したおおざっぱな機能単位で設計者およびプログラマに割り振られ，個別に開発が行われた．この方法は，開発・保守（改造）工数を膨大にする次のような特徴を持つ．

- 同じ処理が複数プログラムに散在する：同一処理であっても別々の開発者に振り分けられると，個別に開発が進んでしまう．そこに変更があると，そのたびに変更の ‘横展開’ が必要になる．
- 設計書がソースコードと乖離する：自然言語で書かれた設計書の変更の実態は，文章の変更・挿入作業であり，生産性においてソースコード変更と同等の工数がかかる．したがってソースコードの変更があったとき，設計書にまで遡って修正されることは少なく，設計書は劣化していく．逆流を嫌うウォータフォール型プロセスがそれを助長する．

このようなウォータフォール型プロセスの弱点を克服するため，アジャイル・ソフトウェア開発の技法が提案され，適用が広まっている．それは，顧客（プロダクト・オーナ）と協業しながら，変化する現場の要求を，繰返しによって柔軟に取り入れていこうとするものである．

しかし，実務ではとにかく短納期に対応せざるをえないために，開発ドメインの簡潔な概念モデル，モジュール化され再利用性・拡張性を保証した実装モデル，実装モデルから導かれる可読性の高いソースコードなどというソフトウェア開発の原則を逸脱せざるをえず，よってそれらの成果物が知識や素材として蓄積されにくい状況にある．実際，それらが他の類似の開発プロジェクトにおいて再利用されることは稀である．またその原則についても，教育や書籍で語られるものは，あくまでサンプル，教科書のレベルを出ていない．実務に直結した題材を扱い，かつ実装による確実な動作実証を行った例はきわめて少ない．

本論文は，この現状をふまえ，標準化され，変化に対応できるアプリケーションのあるべき姿の 1 つを，実装例を含めて実現性を明確にした形で示すことを目的とする．

2. 研究の概要

2.1 これまでの取り組み

変化に強い構造を持った標準化されたアプリケーションを作成するための試みは過去に幾度も行われている．個々の企業活動を，概念モデルを使って情報システムに写像し，パタンとして再利用可能とする試みは，長年の活動の成果として実現しつつあった（たとえば，文献 [3]）．しかし，その努力を知識として共有するためには，大きな障害があった．モデル化を追求すれば，そこに企業秘密としての戦略までがモデル化されてしまう．それはモデル化された企業が著作権の形で保持するために，公開して共有し，改良していくことは不可能となる．

先行研究においてもモデルベース開発とその有用性が述べられてはいるが，そこで取り上げられている題材は「流通業」（文献 [9]）という特定の「業種」であったり，「アプリケーション共通基盤」（文献 [10]）のような業務に関係のない共通部分にとどまっている．それは，前記の理由により，ベンダに業務の知識が蓄積されておらず，したがってそれ以上踏み込むことが困難であったことも原因の 1 つであろうと推測する．前記文献には業務の概念モデルは書かれていない．

本論文では，業務の概念モデルそのものを直接とりあげてその構造を議論する．業務をそのままモデルに写像し，それぞれ異なると思われている企業活動自体に共通性のある構造を見い出し，複数例の業務モデルを研究成果として例示する．これらの概念モデルは，直接的に業務知識そのものであるが，著作者を明示する限りにおいて，第三者が自由に利用してもよいものとする[*1]．

2.2 課題の認識

まず，ここで対象とするアプリケーションは企業の販売管理，生産管理，調達管理といった実行業務を行うものであり，集計や統計の業務ではない．Plan/Do/Check/Action が管理の基本とすれば，実行業務では，活動の予定（Plan）を立て，その実績（Do）を記録することが必須である．このようなアプリケーションが膨大に開発されるので，これに焦点を当てる．さらに，企業活動（ビジネス）を，「契約に基づく資産の移動である」と定義すれば，それらは商流，物流，金流（決済）のビジネスにおけるほとんどの活動を記述できる．これをソフトウェアに写像したものを標準構造として整備し，企業活動の変化に強いソフトウェア構造を実現すればよい．

そこで我々は消費税計算を題材にモデル化を行い，そのモデルに基づいて試実装することで，変化に強い実行業務アプリケーションの標準構造を示す（第 1 ステップ）こととした．消費税計算は販売管理，調達管理システムで必須の代金計算機能の一部である．代金計算はあらゆる企業の基本活動である．また消費税額は商品の基本価格から，後述する複雑なルールにより計算される金流計算の代表的項目である．消費税計算を含む代金計算をアプリケーション

[*1] 本論文で取り上げたすべてのモデル，ユースケース記述，およびコードを，github（https://github.com/kkhn/tax-lib）で公開している．

として開発すれば，商流，金流の 2 つの活動を記録できる．しかも消費税に企業戦略の余地はない．法律で決定された制度であり，会計制度での扱いも明確であるため，そのモデルは共有可能だからである．そして，まさに変化のときを迎えている．5%から 8%，10%への税率変更が決定している．いずれ税率が 10%に改定されるときには，おそらく軽減税率が導入されるだろう．対象は生活必需品に限定されると見られるが，具体的にどの商品かは未定である．そのため決定には多くの論争が予想され，10%適用直前に決定される可能性が高い．

次に，標準構造が他の業務のアプリケーションに適用できるかを検証するために，2 つの応用例をあげる（第 2 ステップ）．その例は，企業が厚労省の指示により必ず実施する年末調整における扶養控除額計算，および同様に実施する健康保険業務である．これらは人事給与システムの一機能であり，販売または調達管理の一機能である消費税計算とは異なるが，消費税計算モデルにあてはめると業務がうまく説明でき，代金計算アプリケーションが標準構造を持つことが証明できると考えたからである．

2.3 研究の目的

アプリケーションの標準構造を，消費税計算を題材とした検討を通して提示することを目的とする．その標準構造は変化に強いものでなければならず，制度や規制などのルール記述の部分とそのルールによって制御される企業活動の部分が分離された概念モデルになると想定する．具体的には，消費税計算ルールや税率決定などの変動要素と，商流，金流という企業活動の不変要素を分離した概念モデル，実装モデルとして提示される．ユースケース記述とコードは実装が可能であることの証拠として提示される．

当該の標準構造が普遍的なものであることは，最初の消費税計算概念モデルを 2 件の応用例に適用することにより検証する．ただし，応用例では各ドメインの概念モデルの作成と評価にとどめる．これらの実装モデルおよびコードについては，別途詳細な検証が必要である．

2.4 研究設問

上記目的を達成できる概念モデル，実装モデル，ソースコードが存在することを示す．ここで，達成しているとは，次の要件を満たしていることをいう．

- 消費税計算の概念モデルは標準構造を用いて表現でき，その構造によって，変化に強く，予測される消費税制と要求の変動と実務の変動に耐えられると推定できること．そのことがモデルから導かれる実装モデル，コード，テスト結果により示されていること．これらが実際の消費税率変動時のユースケースを，アンチシナリオを含めて記述し，テスト結果を検証することで実証されること．
- 実装モデルは概念モデルの基本構造を継承し，モジュール化の原則（凝集性と結合度）が保持され，コードとの対応が容易であること．コードは可読性が高く，ひと目で処理内容が理解できること．
- 他の業務ドメインに対して，消費税計算モデルを応用できることが示されること．

2.5 研究の方法

第 1 ステップでは，ルールと企業活動の分離がどのように設計され実装されていくかについて設計および実装作業を参与観察し，記録した．

第 2 ステップでは概念モデルの洗練と応用モデルの作成を行った．

作業は概略，以下のように進めた．

2.5.1 消費税制度の調査

消費税計算の概念モデル化にあたって，消費税制，会計（一般会計と税務会計）実務および取引実務を調査した．

2.5.2 消費税計算概念モデルの作成

調査に基づきルールを整理し，概念モデル [1], [2] に取り込んだ [7], [8]．消費税率を決定するルールである社内・社外取引の扱い，国内・国外（免税を含む）取引の扱い，課税品目（一般税率，個別税率，軽減税率）・非課税品目の区別，計上日付（仕入/販売とその返品）の扱い，端数まるめルール，総額表示か明細表示か，外税計算をする場合（おもに企業間取引）と内税計算をする場合（おもに消費者との取引）などを盛り込んだ初期モデルを作成した．

2.5.3 消費税計算モデルの洗練

そのうえで企業活動である商流，金流の一般モデルと消費税計算部分を分離し，消費税計算部分を別ライブラリとした．一般モデルとして勘定パタン [3] を導入した．契約と品目（資産）の概念を導入し，前記の計算ルールを企業間の契約によるもの（端数まるめルール，明細単位計算か合計計算表示か），取引種別によるもの（社外・社内，国内・国外取引），品目によるもの（課税・非課税，一般税率・個別税率・軽減税率）を整理してモデルの各部分に配置した．さらに洗練を重ね，消費税計算に照準を合わせた標準構造を持つアプリケーションの概念モデル第 1 版を完成させた．

2.5.4 消費税計算ユースケースの記述

次に業務機能をユースケース記述で表現した．「契約を記録する」および「注文を記録する」のユースケースである．

「契約を記録する」では税の明細単位計算か合計計算か，さらにこの両方に違算が出たときの対処と税額端数のまるめ方法（切り捨て，四捨五入，切り上げ）を，新しく販売契約を締結した顧客との契約条項としてアクタ（たとえば，契約担当者）が入力して記録する．

「注文を記録する」では，契約している顧客からの注文をアクタが入力し，システムに登録されている商品代金か

ら消費税を計算し，表示し，記録する．ここでは，取引のあった日の税率により，消費税が契約にある“まるめ方法”に従ってまるめられていることが確認できるようにすることで2つのユースケース間の整合をとるようにしている．

2.5.5 消費税計算テストシナリオの作成

アンチシナリオとして，一般税率の変更日を事前に設定しておき，その以前の日を取引した日とする入力，以降の日を取引した日とする入力，さらに変更日以前を取引した日とした注文記録を，変更日以降に取り消す（この場合はマイナスの注文として，変更日以前の税率で計算されなければならない）というシナリオを準備した．このシナリオにより，開発したアプリケーションが消費税制の変動とそれにともなう実務の変動に耐えられることを検証する．このようなシナリオに対しては，期間を持った税率の概念だけでなく，取引データも時間の概念が概念モデルに取り込まれている必要がある．取引データには，それを記録した日ではなく取引した日が基準になることも概念モデルで明確に記してある必要がある．

2.5.6 消費税計算実装モデルの作成

次に概念モデルとユースケース記述を基に実装モデルを作成した．実装モデルとその解説は3.2.1項に示す．ただし，軽減税率や特に内税計算については試実装の対象外とした．

2.5.7 テスト駆動によるコードの作成

実装では，シナリオに合わせ，テストコードを書くことから始め，画面とのインタフェース，永続化機能を付け加えた．

2.5.8 消費税計算概念モデルの改善

第2ステップでは，上記の消費税計算モデルのユースケース記述で実装時にスコープ外とした部分を見直した．

結果として「個別契約」という新規概念を導入し，元のモデルにあった「基本契約」と合わせて「契約」という概念を洗練した．すなわち，個々の注文は「個別契約」であり，企業活動はその注文により規定されることを明らかにした．また，契約と品目の2つの概念の中にルールを閉じ込めた．これにより企業活動の変動部分をさらに狭い範囲に固定できた．

2.5.9 応用モデル作成のための調査と選択

この改訂消費税計算モデルの構造が普遍的であることを確認するため，公表できるドメインを捜し．消費税計算モデルの構造にあてはめ，モデル化した．モデル化するドメインは年末調整のうちの扶養控除額の計算，および健康保険は家族に医療費の軽減権を与える（健康保険証はそのviewである）処理である．選択理由は消費税と同様，企業秘密が含まれる余地のないドメインであるからである．

2.5.10 扶養控除額計算概念モデルの作成

扶養控除額計算は登録された扶養家族（家族のうち従業員の給与で生計を立てているもの．本人を含む）の情報に基づき所得税計算のための給与控除額を計算するもので，翌年1月の給与計算に間に合わせるために前年11月頃に該当年の情報を収集する．これを応用例としてモデル化した．

2.5.11 健康保険概念モデルの作成

健康保険は登録された扶養家族（家族のうち従業員の給与で生計を立てているもの．本人を含む．健康保険の対象となりうる家族の範囲は，扶養控除計算の範囲の一部である）の情報に基づき医療費の負担分を決定し（現在，被保険者3割，家族3割），医療費負担軽減の権利を与えるものである．これは対象家族に変動があった場合，その時点で従業員が登録変更を行い，行われた時点で，権利が付与されて健康保険証が発行され，それ以降の医療費額と負担額を計算して給与明細に表示する．これを2つ目の応用例としてモデル化した．

3. 結果

作成されたモデルとコードを以下に示す．なお，コードは試実装にとどまることに注意されたい．

3.1 概念モデル

概念モデルを**図1**に示し，以下に概説する．

3.1.1 業務ロジック部分

概念モデルの中心は業務ロジック，すなわち，消費税計算を発生させる売買取引を標準化した商流部分，および商品代金とその消費税を記録する金流部分である．この基本構造には勘定パタン[3]が使われている．勘定パタンは，取引（Transaction），移動（Entry），勘定（Account）の3つのクラスからなる構造を持っている．後述するユースケース記述の中では，これを“TEA”と省略して用いている．

“TEA”は取引の種別と取引対象の資産，その資産の移動量（数量と金額），移動日を表したものであり，企業活動の標準構造である．資産を「勘定」で表すことで，会計業務への連携を容易にしている．

これらの活動を制御する制度や規制などのルールは，契約と品目という概念に記述される．すなわち，契約クラスの消費税条項の値と，品目クラスとそのサブクラスが，消費税計算の税率とまるめ方法，明細単位計算か合計単位計算かを指定する．この情報は合計単位計算の場合商流取引オブジェクトから直接，または明細単位計算の場合は商流移動オブジェクトを通じて代金取引オブジェクトに渡され，代金取引オブジェクトが消費税計算を起動する．

3.1.2 消費税ライブラリ

消費税ライブラリは，消費税計算をアプリケーションから独立させるため，計算機能をライブラリとして提供するものである．特徴を以下に示す．

- 期間の概念を導入し，税率の期間変動に対応した．税率の決定要素が税品目，税率種型，期間であることを

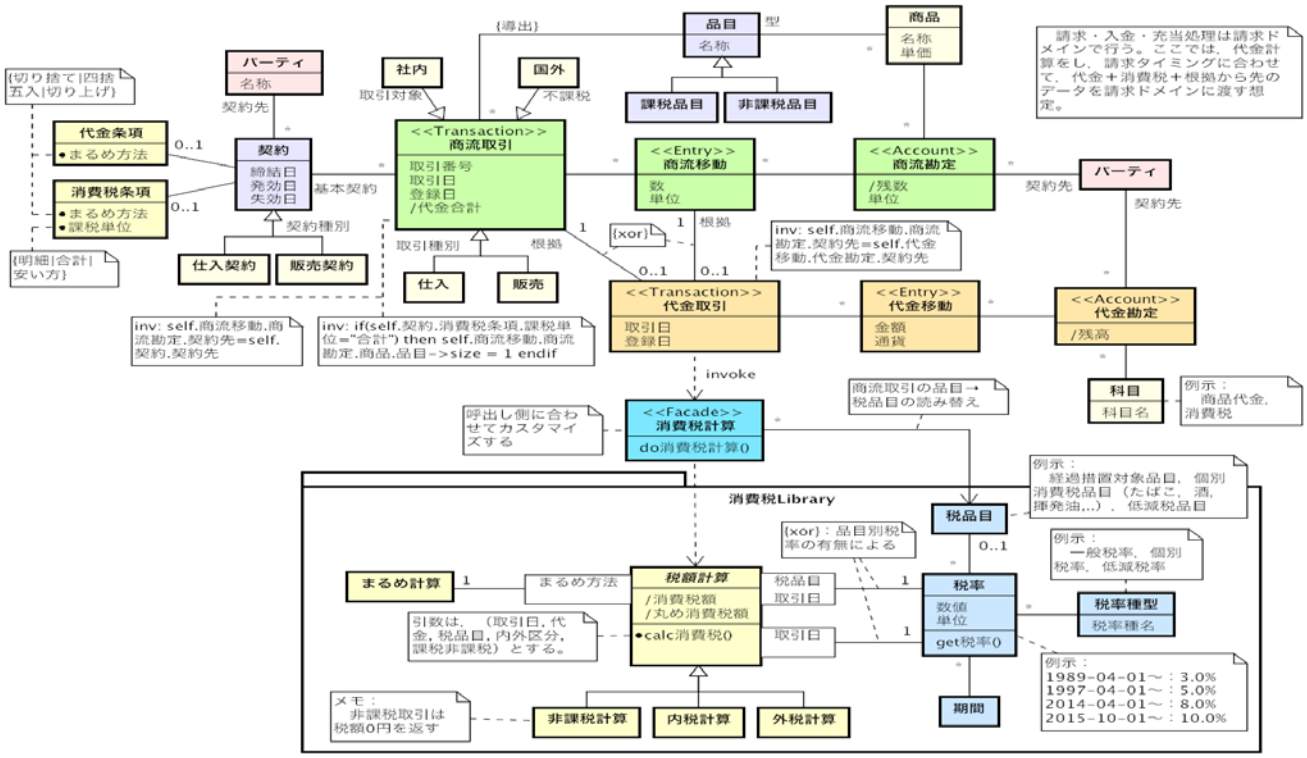

図 1 消費税計算の概念モデル

Fig. 1 The conceptual model of consumption tax calculation.

示している．

- 税額計算クラスを抽象クラスとし，非課税，内税，外税などの計算方式の多様性に対応できる．さらに個別税率の拡張などにも対応できる．
- 税額計算は取引日を限定子として税率クラスと関連している．これにより，期間とは取引日が含まれる期間であることを示している．

3.1.3 消費税計算 Façade

軽減税率など消費税率を決定する品目と企業活動で使用される品目とは概念が異なるため，この対応をとるための処理を消費税計算 Façade で行うこととした．この変換規則は使用システムに合わせて最適にカスタマイズされる．

3.2 実装モデル

実装モデルを**図 2** に示し，以下に概説する．

3.2.1 実装モデルの解説

- エンティティとして永続化されるものには基本的にすべて識別子を明示した．識別子はクラス内の属性表示中の破線から上の属性である．エンティティとは離散的にこの世の中に存在するモノや事象を表し，それらは識別されるべきである [6]．
- 消費税計算ロジックについては，唯一のインスタンスに計算操作を持たせるよう Singlton パタン [4] を適用した．
- 非課税・内税・外税計算（CalculateConsumptionTax）とまるめ計算（RoundingAmount）にはそれぞれ Strategy パタン [4] を適用した．
- 税率計算クラス（TaxRate）は列挙型とし，税率を表引きできるようにした．例示に属性値を税制表記している．

3.2.2 ユースケース記述

ユースケース「注文を記録する」の記述の一部を**表 1** に示す．

3.2.3 テストシナリオ

テストシナリオは，ユースケース記述の一部として記述され，ユースケースの理解とテスト駆動開発用のテストケースとして利用される．その一部を**表 2** に示す．

3.2.4 コード

次に実装モデルに基づいて作成されたコードを示す．コード全体の構造は，MVC の構造 [5] となっている．言語は Java であり Java SE7 に準拠して作成した．開発環境には Eclipse を使用した．以下に示す例では，いわゆる setter/getter を省略している．

(1) コード例 1：ConsumptionTaxCalculator.java

消費税計算（CalculationConsumptionTax）およびまるめ計算（RoundingAmount）の条件による振舞いの違いを Strategy パタンで実装している．

```
public class ConsumptionTaxCalculator {

    private BigDecimal price;
    private Date effectiveDate;
    private TaxItem item;
    private boolean isTaxed;
```

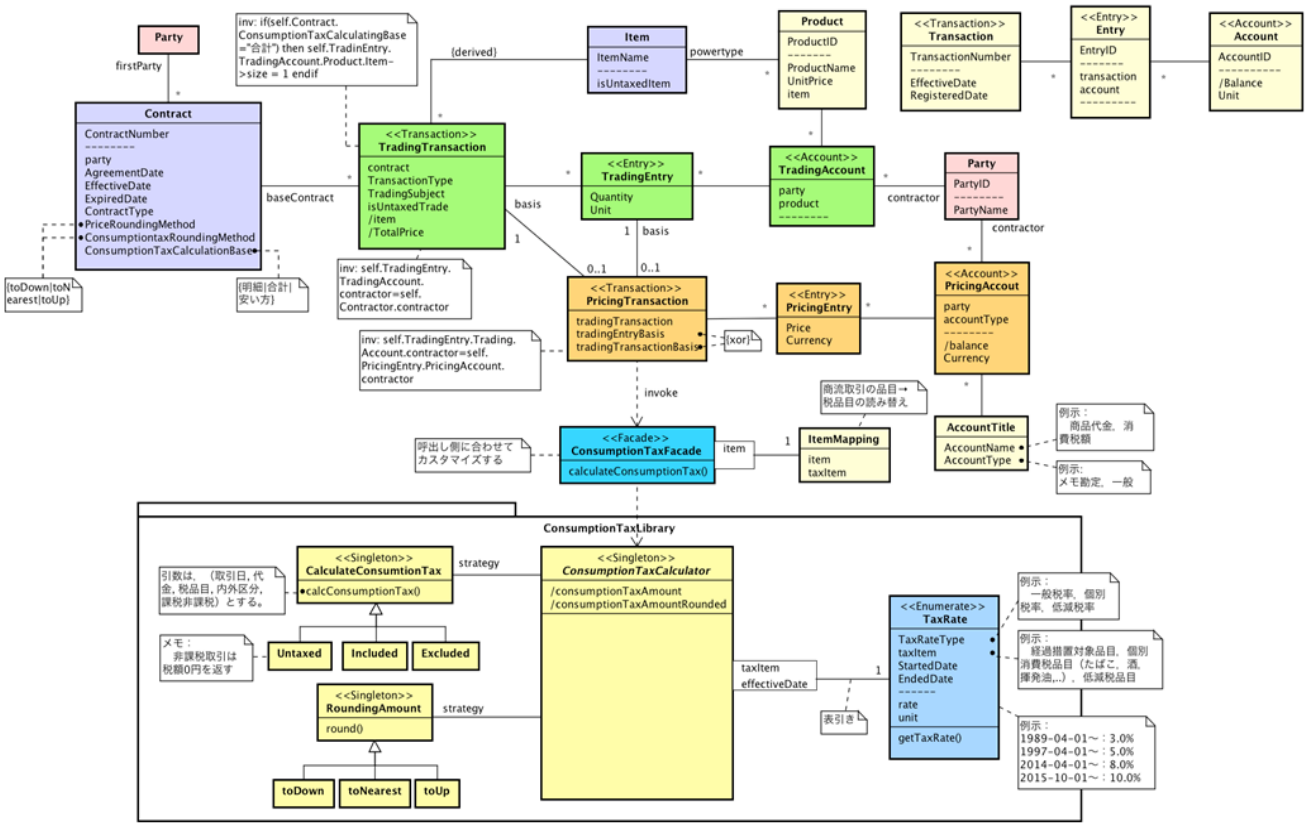

図 2 消費税計算の実装モデル

Fig. 2 The implement model of consumption tax calculation.

表 1 ユースケース記述「注文を記録する」の一部

Table 1 A part of the was case description “Record the Order”.

2. 注文を記録する

アクタ：営業
概要： 取引先からの注文内容を基に，代金計算を行った上で妥当な注文を記録する．
目的： 適切に出荷し，代金を回収したい．
事前条件：商品がある．商流勘定と代金勘定は，すでに存在している場合がある．
事後条件：商流取引，商流移動がある．商流勘定がなかった場合は，それが新たに作られている．これに関連して，代金取引，代金移動がある．代金勘定がなかった場合は，それが新たに作られている．
基本系列：
①アクタはこのユースケースを起動する．
②システムは，注文内容（取引種別［販売］，不課税取引区分，取引対象区分，商品，数，取引先，取引日）の提示を求める．
③アクタはそれらを提示する．上記注文内容はすべて必須項目．
④システムは，<<システムの振る舞い>>--
⑤アクタは確認する．
⑥システムは商流取引と代金取引（各 TEA）を永続化する．
代替系列：
＜省略＞
備考：
＜省略＞

表 2 テストシナリオの一部

Table 2 A part of the testing scenario.

シナリオ：
①営業の山田直子は，7 月 30 日に麺屋海神から，鯛のアラ（課税品目）5.5 キログラム（273 円/キロ）とヒラマサのアラ(課税品目)8.3 キログラム（126 円/キロ）の注文を受け取り，販売取引，課税，社外取引として，その内容を記録した．代金計算の結果が，鯛のアラ 1501 円，ヒラマサのアラ 1045 円，消費税率は 5%, 販売契約に基づき消費税は 127 円と表示され，これが正しかったので確認した．
②営業の山田直子は，8 月 1 日にも麺屋海神から，鯛のアラ 5.5 キログラムとヒラマサのアラ 8.3 キログラムの注文を受け取り，その内容を記録した．代金計算の結果が，鯛のアラ 1501 円，ヒラマサのアラ 1045 円，消費税率は 8 月 1 日から 8%になったので，消費税は 203 円と表示され，これを確認した．

```
    private CalculateConsumptionTax calculator;
    private RoundingAmount rounder;

    public ConsumptionTaxCalculator(
        BigDecimal price,
        Date effectiveDate,
        TaxItem item,
        Boolean isTaxed,
        CalculateConsumptionTax calculator,
        RoundingAmount rounder) {
      this.price = price;
      this.effectiveDate = effectiveDate;
      this.item = item;
      this.isTaxed = isTaxed;
      this.calculator = calculator;
      this.rounder = rounder;
    }

    public BigDecimal getConsumptionTaxAmount() {
      return calculator.calcConsumptionTax(effectiveDate,
```

```
            price, item, isTaxed);
    }

    public BigDecimal getConsumptionTaxAmountRounded() {
     return rounder.round(getConsumptionTaxAmount());
    }
}
```

(2) コード例 2：CalculateConsumptionTax.java

外税，内税，非課税ごとに消費税計算ロジックを用意する．Singleton の指定に対しては Java の enum 型を用いることで実現している．

```
public enum CalculateConsumptionTax {
  外税 {
    @Override
    public BigDecimal calcConsumptionTax(Date effectiveDate,
        BigDecimal price, TaxItem item, boolean isTaxed) {
      TaxRate rate = findRate(item, effectiveDate);
      BigDecimal number = BigDecimal.valueOf(rate.getRate());
      BigDecimal p = BigDecimal.valueOf(100);
      return price.multiply(number).divide(p);
    }
  },
  内税 {
        // スコープ外
      }
  },
  非課税 {
        // スコープ外
  };

  public TaxRate findRate(TaxItem item, Date effectiveDate) {
    return TaxRate.getTaxRate(item, effectiveDate);
  }

  public abstract BigDecimal calcConsumptionTax(
      Date effectiveDate,
      BigDecimal price,
      TaxItem item,
      boolean isTaxed);
}
```

(3) コード例 3：TaxRate.java

税率のクラス．実装モデルでは将来の拡張を考慮し，列挙型として表引きを行う指定になっているが，現実には税品目ごとの税率が定められていない．ここでは実装判断に基づき，単純な期間別の税率を定義した．

```
public class TaxRate {
  private TaxRateType type;
  private TaxItem taxItem;
  private Date startedDate;
  private Date endedDate;
  private int rate;
  private Unit unit;

  public static List<TaxRate>
      $instances = new ArrayList<TaxRate>();

  public TaxRate(TaxRateType type,
      TaxItem item,
      Date startedDate,
      Date endedDate,
      int rate,
      Unit unit) {
    this.type = type;
    this.taxItem = item;
    this.startedDate = startedDate;
    this.endedDate = endedDate;
    this.rate = rate;
    this.unit = unit;
  }

  public static TaxRate getTaxRate(TaxItem item, Date date) {
    for (TaxRate t : $instances) {
      if (t.getTaxItem() == item &&
          t.getStartedDate().compareTo(date) <= 0 &&
            t.getEndedDate().compareTo(date) >= 0 ) {
        return t;
      }
    }
    return null;
  }
}
```

3.2.5 動作の確認

コードが設計どおりに稼働していることをテストツール JUnit4 の実行結果で確認する．**図 3**，**図 4**，**図 5** に，テストツールのコンソール出力の当該部分をキャプチャしたものを示す．テストケースは，上記ユースケースシナリオで一部を例示したものである．

図 3 は 7 月 30 日に注文を受けた結果である．図 4 は，消費税が 8 月 1 日に 5%から 8%に改定されたことを仮定して前後の注文入力を実行している．消費税率が変わり，税額が計算されている．複数明細の合計で計算していることにも着目されたい．図 5 は，7 月 30 日に取引した注文を 8 月の消費税率変更後に取り消している．このときの税率は 5%で再計算されている．複数明細の一部訂正であるため，この場合遡及計算が行われる．

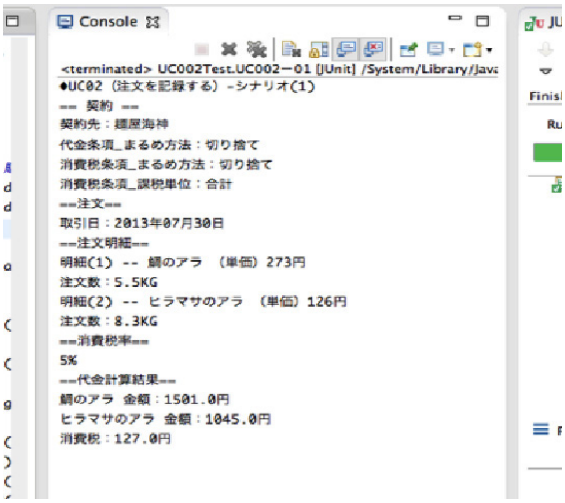

図 3 テスト結果 1：7 月 30 日取引の注文
Fig. 3 Test Result 1: Order traded on July 30.

3.3 消費税計算モデルの応用

3.3.1 消費税計算モデルの洗練

第 1 ステップの概念モデルには，商流取引と代金取引をまとめる「注文」の概念が希薄だったので，以下の改善を加えてモデルを改訂した（図 6）．

- いわゆる「注文」を個別契約ととらえ，基本契約クラスに複数の個別契約クラスを持たせた．
- 商流取引を商品取引に改称したうえで，個別契約クラスが商品取引，代金取引クラスをまとめることとし，かつ代金勘定オブジェクトを作成するためのルール（仕訳ルールと呼んでもよい）をここに持たせた．消費税を明細ごとに計算するか，取引全体の合計金額に税率を掛けるか，または両方計算しておいてその安い方を採用するかというルールである．このルールにより明細ごとに消費税を計算するときは消費税オブジェク

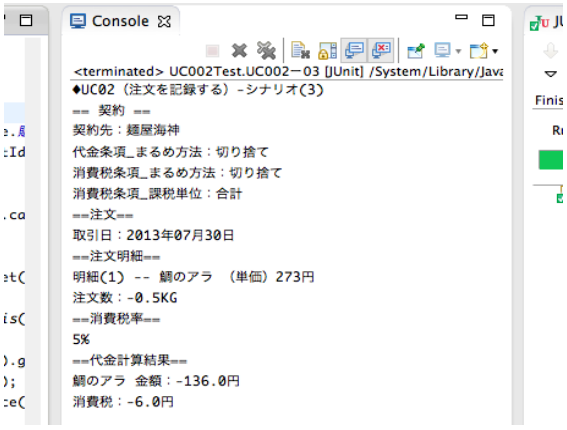

図 4 テスト結果 2：8 月 1 日取引の注文

Fig. 4 Test Result 2: Order traded on Aug. 1.

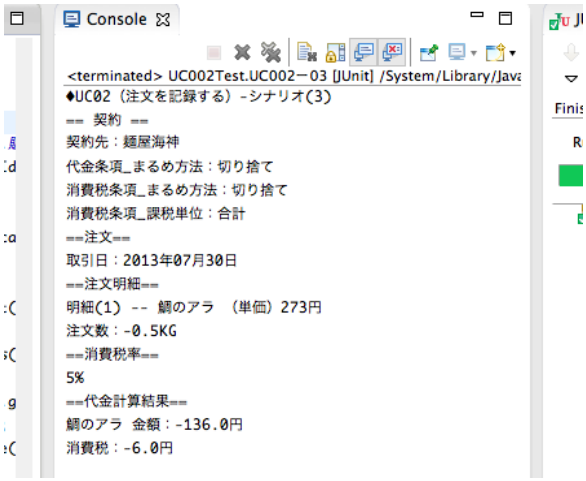

図 5 テスト結果 3：7 月 30 日注文の遡及訂正

Fig. 5 Test Result 3: Retroactive correction for the order on July 30.

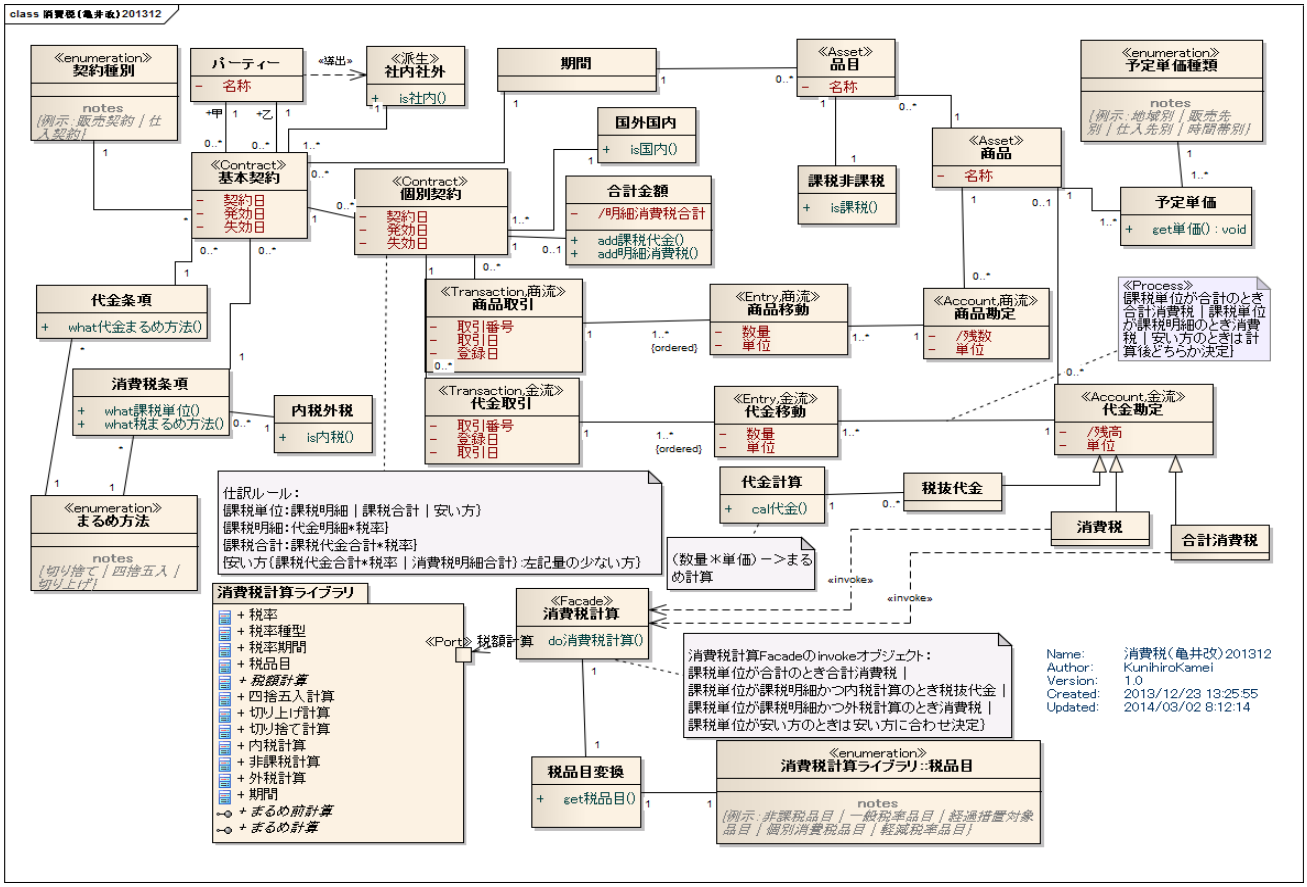

図 6 改訂消費税計算モデル

Fig. 6 Reused conceptual model of consumption tax.

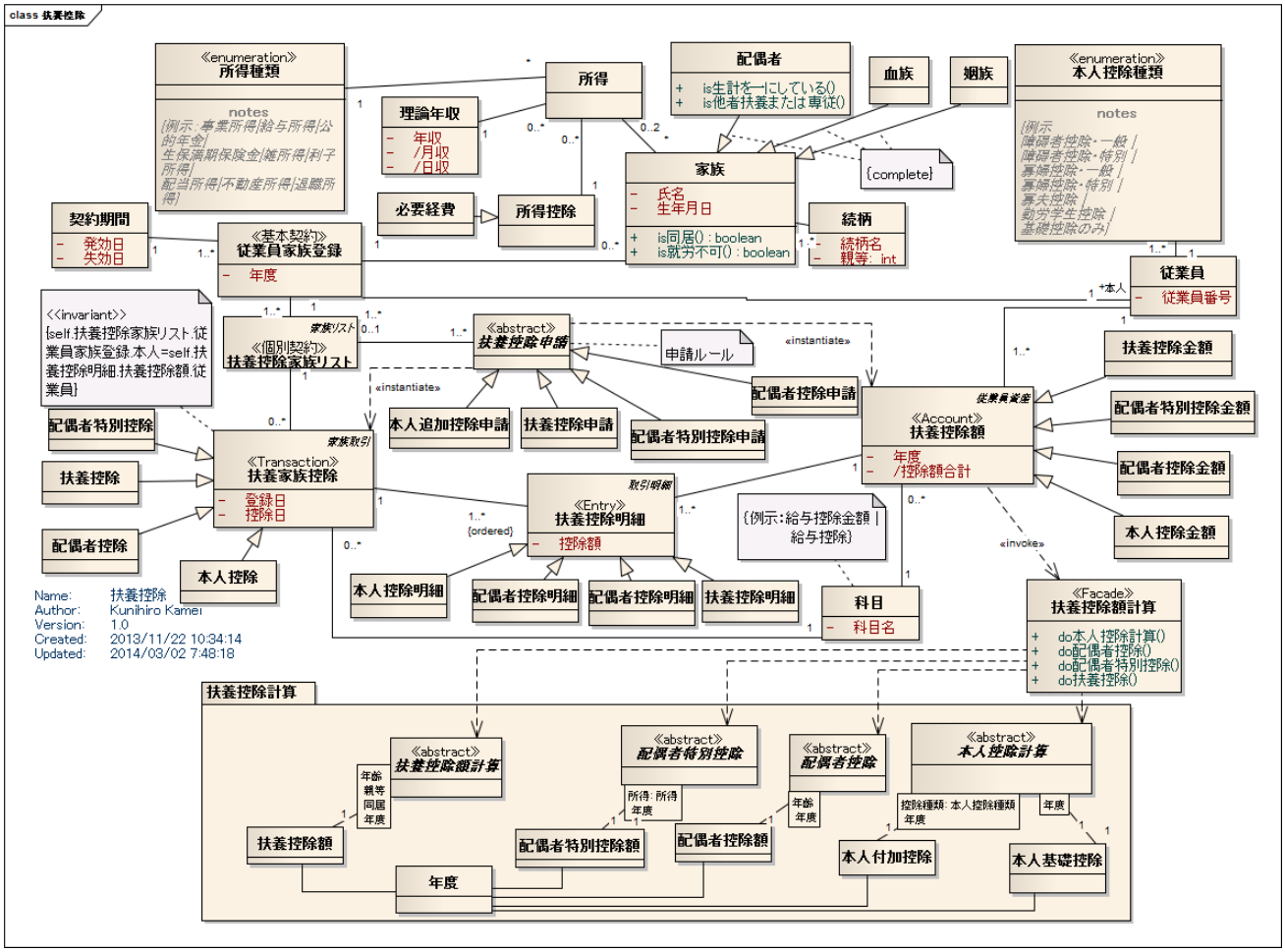

図 7 扶養控除額計算モデル

Fig. 7 The conceptual model of dependency exemption calculation.

トを，合計で行うときは合計消費税オブジェクトを生成する．これは第 1 ステップでの「注文を記録する」ユースケースの基本系列⑤，⑥（表 1 の下線部）を実現するためである．

- 代金勘定クラスから消費税計算ライブラリを呼ぶこととした．
- 基本契約と品目に期間の概念を付加した．品目に期間の概念を付与しておくと，税率 8%時一般税率であった品目が 10%増税時以降軽減税率となるという管理ができる．これによりこのような品目についても増税時をまたがった遡及入力を可能とする．

3.3.2 扶養控除額計算モデル

消費税計算モデルを基に扶養控除額計算モデルを作成する（図 7）．家族の関係は本人からの親等と続柄で表現されるので，これをモデル化したものを家族クラスとする．基本的に同居している親族は対象となる（別居家族は送金証明があること）ので，この家族の親等，続柄を登録しておく．控除対象かどうかは，所得が一定以上あることと年齢で決まるので，これを登録する．

この従業員家族登録を所属企業と従業員の基本契約とする（所属企業はモデル上表現していない）．基本契約には消費税計算の基本契約同様契約期間情報を持ち，家族に変更が出た時点で次の期間の契約を作る仕組みにして，遡及入力や予定計算を可能とする．家族情報を変更した時点の扶養控除対象家族のリストが個別契約である．対象家族には年齢（16 歳以上），所得（基本的に 38 万円以下）の制限があるのでこの制限内の家族をリストしたものがその年度における扶養控除の個別契約である．個別契約には消費税計算モデルと同様対象者リストの内容によって本人控除・配偶者控除・配偶者特別控除・扶養控除のそれぞれの控除額を計算し，記録するルールを保持する．

各控除は TEA で表現でき，扶養控除計算ライブラリは各年度における控除額の計算テーブルを提供して主に所得額によって変わる控除額を年度に合わせ計算できるようにしている．

3.3.3 健康保険モデル

健康保険モデルは，扶養控除額計算モデルと同様に従業員家族登録が基本契約である．健康保険の対象家族は扶養控除の対象家族のサブセットであり，基本契約として同じモデルを用いることができる．

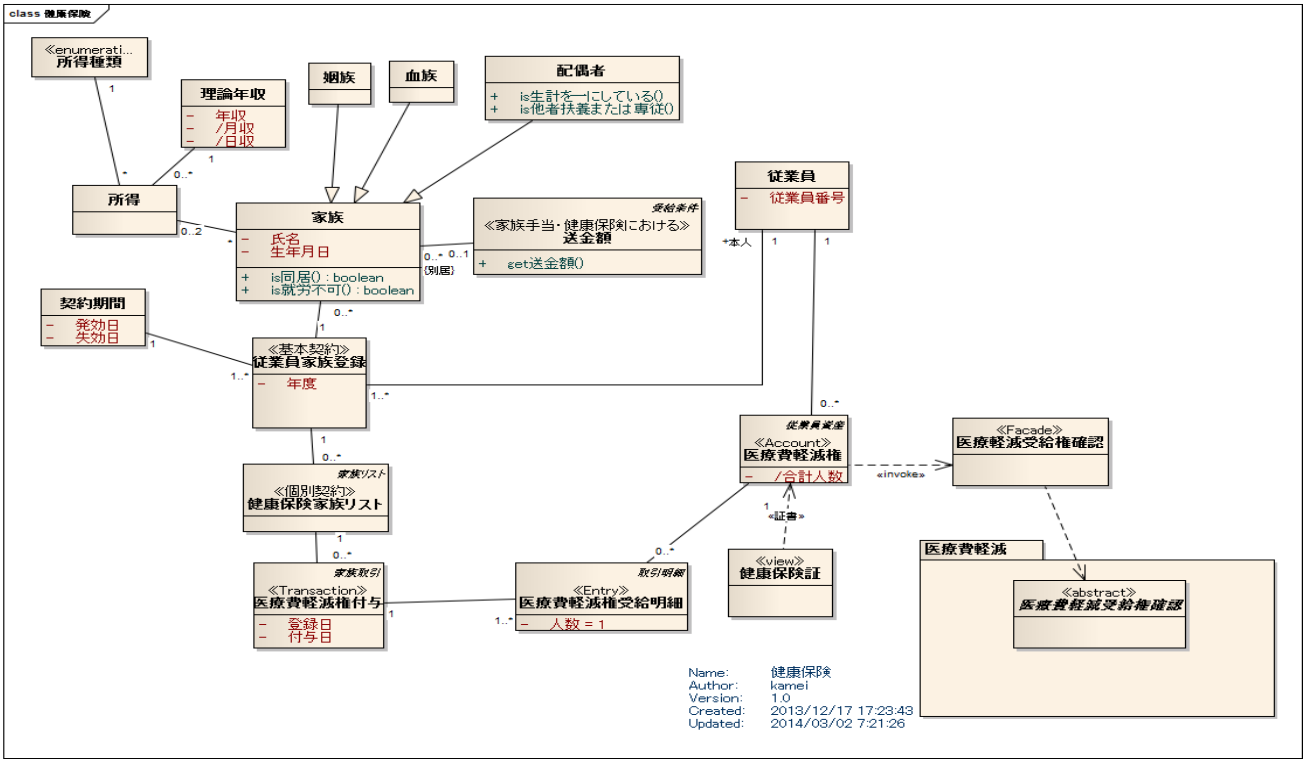

図 8 健康保険モデル

Fig. 8 The conceptual model of health insurance.

個別契約である健康保険家族リストは，基本契約をもとに扶養家族情報変更時に扶養家族対象者リストとは別に作成される．健康保険のモデルでは従業員とその家族に医療費の負担軽減を受ける権利を与えるものであるから，この権利の付与という取引を TEA の構造で記述する（図 8）．

3.4 成果物の評価

消費税計算を題材に取り上げ，さらにそこから想起して汎化と応用を行って概念モデルの例を作成することで，情報システムの現状における課題と解決策の例示ができた．

3.4.1 設計原則への準拠性

ルールと活動の分離，消費税計算部分の分離が行われ，勘定パタンが導入された概念モデルの標準形が得られた．ここから実装モデルおよびコードが導出されている．

3.4.2 概念モデルの良さ

概念モデルは，企業活動を商流・金流の基本部分と消費税計算を明確に分離している．基本部分は構造部品の概念を提唱している．消費税計算部分はライブラリの形で提供される．税率を決める部分は，業界，企業に合わせたカストマイズを可能とする形となっており，このモデルの柔軟性とライブラリの再利用性を高めている．

3.4.3 実装結果

実装モデルは，識別子の導入，Façade，Singleton，Strategy というデザインパタンの適用などの実装モデルの特徴が記された．TaxMapping クラスと TaxRate クラスで消費税カストマイズ部分の構造が明確になった．コーディングはこれを継承している．テストの結果もユースケースの実行が目に見える形で示されている．これは実装の証明を説明するための重要なプラクティスとして評価したい．

3.4.4 消費税計算モデル改版

消費税計算モデルは，個別契約という概念を導入することで次のような利点を得た．

- 消費税計算のルール記述を契約（基本契約と個別契約）と品目に閉じ込めることができた．これは企業活動を変動部分と固定部分に分ける点でさらに進歩している．
- 代金勘定を作成するためのルール（仕訳ルール）を個別契約に記述した．これは企業活動の記録ルールの記述場所を特定することで，さらに構造を標準化し，このモデルの再利用性を高めることとなっている．
- 基本契約と品目に期間の概念を付与することに加え，TEA の TE の部分に企業活動の履歴を記述する部分が含まれていることから，情報システムが陥りがちな欠陥として前記した遡及入力ができない，予定計算ができない，活動履歴を照会できないなどの事項を構造的に防いでいる．

3.4.5 扶養控除計算・健康保険モデル

これらのモデルは人事給与システムの機能をモデリングしたもので，販売管理あるいは調達管理の一部分である消費税計算とは，概念やドメイン知識が異なると思われるが，同様の構造を応用してモデリングが可能であることが

分かった．このことは消費税計算モデルの普遍性の高さを示すものとして評価できる．

3.4.6 課題

第 1 ステップではその範囲での画面・永続化の実装を行ったが，スコープ外とした部分を含む，個別契約の中に記述する TEA の記録ルールの部分は標準処理を作っておくべきであった．概念モデルの進化と実装モデルの進化は同期しなければならない．また，標準構造開発のためには，消費税計算のモデルからまた多くの応用モデルを作成して検証する必要がある．応用モデルについてもユースケースや実装による設計の揺さぶりを行って，洗練する必要がある．

4. 結論

本研究の目的である「変化に強い標準化されたアプリケーション構造」を達成できる概念モデル，実装モデル，ソースコードが得られた．具体的に「2.4 研究設問」であげた要件に対する評価は次のとおりである．

- 提示した概念モデルは，標準構造を用いて表現され，調査した範囲での消費税計算仕様はすべて概念モデルに盛り込まれた．消費税計算ルールは契約と品目クラスに定義され，税率決定ロジックと税額計算ロジックは消費税ライブラリに集められている．その実装は，予測される消費税制と実務の変動に耐えられた．それはユースケース記述とその実現により検証されている．
- 実装モデルは，概念モデルの基本構造を継承し，モジュール構造はデザインパタンを持ち込むことで，モジュール化の原則を守るように努められた．コードは，実装モデルとの対応によって，可読性が高められている．
- 消費税計算の応用例として作成した扶養控除額計算と健康保険の 2 つの概念モデルは業務適合性のあるモデルといえる．このモデルを作成した筆者の 1 人は，業務改善提案の一環として，このモデルから導き出された扶養家族の登録ルールをルール説明書の改善案として顧客に提出している．

5. おわりに

本研究では，一部の範囲ではあるが，変化部分と固定部分の分離された変化に強い標準的構造の一例を示し，それがアプリケーションとして実装可能なことを示した．情報システム開発における大規模，短納期化への対応は，ここに示したような概念モデルによって表現された業務知識，勘定パタンのような構造部品，消費税ライブラリのような機能部品の再利用によって行われるべきである．

しかし，ここであげた概念モデルおよびその実装は，試行レベルにすぎない．現実社会への適用性の判定までには，本格運用とフィードバックの何度かの繰返しが必要である．試実装を本格運用に向かわせるためには多くの課題があることを理解している．本論文での具体的な成果物の提示が，そうした課題の解決に少しでも貢献できれば幸いである．

参考文献

[1] 児玉公信：UML モデリング入門，日経 BP (2008).
[2] 児玉公信：UML モデリングの本質，日経 BP (2011).
[3] Fowler, M.（著），堀内 一（監訳）：アナリシスパターン，ピアソンエデュケーション (1998).
[4] Gamma, E. et al.（著），吉田和樹ほか（訳）：オブジェクト指向における再利用のためのデザインパターン（改訂版），ソフトバンククリエイティブ (1999).
[5] Buschmann, F. et al.（著），金沢典子ほか（訳）：ソフトウェアアーキテクチャ—ソフトウェア開発のためのパターン体系，近代科学社 (2000).
[6] Evans, E.（著），今関 剛（監訳）：ドメイン駆動設計，翔泳社 (2011).
[7] 亀井邦裕ほか：消費税計算のための概念モデルの作成と試実装，情報処理学会研究報告，Vol.2013-IS-125, No.3 (2013).
[8] 亀井邦裕ほか：消費税計算のための概念モデルとその継承，情報処理学会研究報告，Vol.2014-IS-127, No.2 (2014).
[9] 斉藤康彦：問題領域基本モデルを活用した企業情報システムの開発手順，北海道大学紀要，Vol.15, No.1, pp.39–50 (2003).
[10] 石原 鑑ほか：企業情報システム再利用開発のためのアプリケーション共通基盤のモデルベース設計，電学論 C，Vol.129, No.2 (2009).

亀井 邦裕

産業技術大学院大学客員研究員．1979 年京都大学理学部（物理学専攻）卒業．同年富士通（株）入社．以来，流通業・製造業向けアプリケーション開発に従事．2013 年より現職．共訳書に『PMBOK ガイド・マニュアル（第 5 版対応）』（オーム社）等がある．

児玉 公信 （正会員）

（株）情報システム総研取締役副社長．1978 年東京都立大学人文学部（心理学専攻）卒業．石油元売り，北海道大学工学研究科受託研究員，製鉄系情報子会社を経て，2008 年より現職．技術士（情報工学部門），博士（情報学）．著書に『UML モデリング入門』，『UML モデリングの本質（第 2 版）』（ともに日経 BP）ほか，訳書に『リファクタリング（新装版）』（オーム社）等がある．日本心理学会，日本経営情報学会，ACM 各会員，本会シニア会員．

細澤 あゆみ

（株）情報システム総研社員．2011 年静岡県立大学大学院経営情報学研究科修士課程修了．2011 年より現職．修士（経営情報学）．

成田 雅彦

産業技術大学院大学産業技術研究科教授．1980 年早稲田大学大学院数学科修士課程修了．同年富士通（株）入社．以来，OS/ミドルウェアの企画・戦略立案と国際標準活動に従事．2008 年より現職．博士（工学）．著書に『CORBA と Java 分散オブジェクト技術』等がある．電子情報通信学会，精密工学学会，日本ロボット学会，人工知能学会会員．