業務知識のチェック関数化にもとづく業務処理システム構築法の提案

Proposal to construct business processing system based on check function of business knowledge

坂口 直也† 中西義尚‡ 金田重郎‡
Naoya Sakaguchi† Yoshinao Nakanishi‡ Shigeo Kaneda‡

† 同志社大学　理工学部
‡ 同志社大学大学院　工学研究科
† Faculty of Science and Engineering, Doshisha Univ.
‡ Graduate School of Engineering, Doshisha Univ.

要旨

業務処理システムでは，一般に，事務処理を手続き的フローとして実現する．しかし，手続き的フローでは，業務規則が他の業務規則と融合され，業務規則変更時のメンテナンスは容易ではない．そこで，本稿では，業務規則を制約として表現し，制約とオブジェクトのみでシステムを構成する手法を提案する．データが変更されると，システムは制約を用いて，データ状態の整合性をチェックする．不整合が生じた場合には，ユーザが不整合を解消することにより業務処理を実現する．制約は必要十分条件であるため，業務規則の追加・変更・削除は業務規則毎に独立しており，業務規則変更に容易に対処できる．実際に，自動車税業務についてプロトタイプシステムを構築し，ユーザの操作数を従来システムと比較した．

1. はじめに

業務処理システムの開発では，業務手続きを洗い出し，それをシステムに実装する手法がとられる．しかし，業務処理のすべてについて，手続きを網羅的に把握することは非常に困難であり，処理漏れなどがおこる．またその手続きによる処理の結果が，事務規則に合致しているどうか，手続きからでは，判断が困難である．上記問題を解決するため，既に，著者らは業務規則を制約として表現し，データ伝搬による制約再充足問題として業務処理を実現するシステムを提案している[1]．しかし，従来システムは，複雑な機能を持つ制約伝搬エンジンを用いており，必ずしも簡明なシステムではない．

そこで本稿では，業務規則を制約として表現する点は従来システムと同様であるが，データ整合性のチェックのみをシステムが行い，制約の充足処理は，ユーザが行う，より簡明な制約表現に基づく業務システム構成法を提案する．業務規則は，必要かつ十分条件である制約として表現されているため，業務規則変更は，当該制約にしか影響せず，業務規則変更への対応は容易である．また，業務規則を用いたあらゆる処理が可能となるため，レアケースへの対応ができないといった問題も生じない．

以下，第 2 章では制約表現を述べ，第 3 章では，提案システムの概要とシステムの動作を説明する．第 4 章では，自動車税を対象としたプロトタイプの実現結果を紹介する．第 5 章で提案システムの評価を行い，第 6 章はまとめである．

2. 業務知識

本稿では，業務処理において，本質的に手続きの背後には事務規則が存在し，手続きは事務規則を満足し続けるための方法であると考える．業務処理を事務規則の整合性を維持する処理であると考え，事務規則などから，必要十分条件を抽出して，制約として表現する．提案する業務システムではその制約の維持を管理する．制約関係を CSV ファイルなどで，表現しておき，法改正など手続きや制約関係が変化した際にも，制約が書かれているファイルを新しい制約関係に修正を行うことで，簡単に対応が可能である．またこのシステムはユーザが制約の充足処理を行うため，データ項目が制約を充足しているかのみをチェックする．

2.1. 整合性制約

この節では，事務規則を制約に変換するプロセスについて説明する．例えば，事務規則が

- 自動車の所有者は，自動車の所在地と同じ県内に納税管理人がなければならない．
- 上記以外の場合は，都道府県知事に申請して承認が必要である．

が存在した場合，制約は表 1 のようになる．表 1 では，住民クラスに住所という属性，自動車クラスに所在地と承認の属性が存在する．ここでは，住民は自動車の納税管理人としている．

この制約条件に満たさないデータの状態（例：納税管理人は，自動車の所在地と同じ県内に住所にあるが，承認を受けている）などは，事務規則違反であると考える．また，制約条件を複数満たすデータ状態も事務規則違反である．

表 1　自動車における整合性制約

住民	自動車	
住所	所在地	承認
○○府	××県	ある
○○府	○○府	ない

2.2. 業務知識のチェック

2.1 章の制約条件を用いて事務規則の整合性を確認する. 制約条件を満足しないとき, 例えば所有者は, 自動車の所在地と同じ県内に住所があるが，承認を受けているなどといったデータ状態になっている場合，データの整合性が取れておらず，警告として担当者に提示する．解決手段としては，住民の住所が誤っている，自動車の所在地が誤っている，承認が誤っていると 3 通りの解決方法があるが，そのどの方法を行えば制約を満たした状態になるのか自動的に決めることができない．そこで本手法では，担当者がデータを見ることで，誤っている箇所を担当者自身に修正させ，データの整合性制約を解消する．

3. 提案システム

3.1. 提案システム概要

提案システムでは事務規則を分析し，制約を作成する．制約はチェックリストとしてシステム内に取り込み，View 層と比較を行うため，メンテナンスを容易に行うことができる．提案システムは，Model 層・View 層・Controller 層からなる．以下に各層で行われる動作を示す．システムの概要を図 1 に示す．

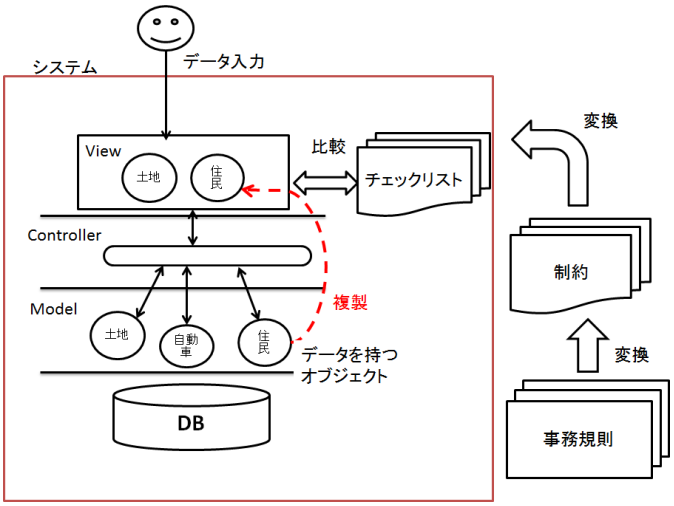

図 1　システム概要

- Model 層 : Model 層のオブジェクトはそれぞれ，DB のデータを保持している．モデル層のオブジェクトに変更があった際に，DB と同期をとる．
- View 層 : ユーザによって，データの変更をしたいオブジェクトのコピーを Model 層のオブジェクトから取り出し，表示を行う．また，ユーザが入力したデータをチェックリストと比較して，整合性制約を確認する整合性制約を充足することを確認してから，Model 層のオブジェクトに変更を適応する．
- Controller 層：Model 層のオブジェクトから必要なオブジェクトを選択し View 層にコピーする．View 層でチェックリストと比較する際に View 層にコピーしていないオブジェクトが必要になれば，Model 層から取り出す．

例えば，表 1 のように，住民の住所は住民のオブジェクト，自動車の所在地と承認については，自動車のオブジェクトの属性として表現されている場合について考える．自動車の属性値変更時にエラーが出た場合，View 層に住民のデータもコピーして表示する必要がある．

3.2. 提案システムの流れ

以下に提案システムの大まかな流れを説明する．

【Step0】業務知識を理解して，事務規則を制約として表現されたファイルを変換する．

【Step1】担当者がオブジェクトの属性を変更しようとした際に，モデル層のオブジェクトをコピーし，オブジェクトの属性すべてをビューに表示する．ここで，担当者が業務処理を理解して，変更する箇所に手を加える．

【Step2】担当者によって手を加えられたデータは，整合性制約を満たさない可能性があるため，整合性の確認を行う．整合性制約の確認には制約と入力データ（必要であれば，モデル層のオブジェクト）を比較する．これによって業務知識をチェックする．

【Step3】制約を満たした場合：入力されたデータを業務規則に合致しているものとして，モデル層のオブジェクトにデータの更新を反映する．

制約を満たさない場合：チェックに使用したデータ項目のどこかに業務規則を満足しない項目があるはずなので，そのデータ項目が含まれるすべてのオブジェクトの属性を全部ビューに表示する．

Step1-3 の手順を繰り返していくことで，業務規則を常に満たす状態にする．

4. 実装

提案システムは C#で作成した．今回提案システムを使用するに当たり，自動車税を用いている．自動車税を用いた理由として，一つ一つのオブジェクトが独立していていること，基本的な動作に集計や統計量が無いことが挙げられる．自動車税については地方税法から制約を作成し，提案システムのプロトタイプを作成した ．

提案システムを実行した際の画面の説明をする．ユーザは各項目にデータを入力する．全て入力し終わった後，画面右上にある比較というボタンを押す．データが正しく入力されていない場合，その下のテキストボックスに『NG』と表示され，エラーがあった場合該当項目と思われる部分が赤くなる．ユーザがデータを正しく入力した場合，その下のテキストボックスに『OK』と表示され，データが正しく入力されたことがわかる．その様子を図 2 に示す．

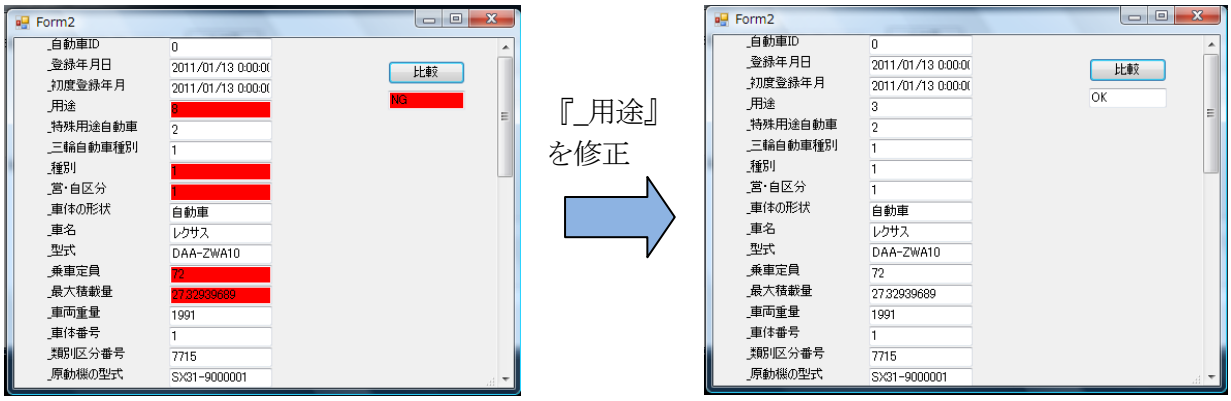

図 2　提案システムの実行画面（データ修正時）

5. 評価

評価実験では，自動車税・自動車取得税申告書を例に挙げ，現行システムと提案システムの入力項目数・ページ数を比較してみた. 現行システムの場合, 入力項目数は 40 個, ページ数は 7 であるのに対し, 提案システムの場合，入力項目数は 47 個，ページ数は 2 である．この結果，入力数はさほど変化がないが，ページ数を削減することに成功している.

また，提案システムでは業務規則を CSV にデータとして表現している．その中身には自動車の種類，自家用／営業用，排気量（最大値・最小値），積載量（最大値・最小値），種別，一般乗り合い用，乗車定員（最大値・最小値），税額が記載されている．例えば，自動車の種類が乗用車であり排気量が 1.8ℓの車の自動車税は，CSV のデータの『自動車の種類が乗用車，排気量が 1.5ℓを超え 2ℓ以下のもの』のところに当てはまるので，年額 9,500 円ということがわかる．このように，提案システムでは CSV で制約を表現している.

提案システムでは業務規則を制約としているので，その変更があった場合は，現行システムでは業務フロー全体を見直す必要があるが, 提案システムではシステムの変更は業務規則である CSV 内のデータの変更のみで済む，という柔軟性を持つ.

6. 終わりに

本稿では，　業務規則を制約として表現し，データ整合性のチェックをシステムが，制約の充足処理はユーザが行うシステム構成法を提案した．評価実験では，データ入力時の手間を省き，業務規則の変更に対し大きな柔軟性を持つことがわかった.

参考文献

[1] Megumi Ishii, Yutaka Sasaki, Shigeo Kaneda: A Constraint Approach for Clerical Works, IEEE Intelligent Systems, pp.64-72, Jan./Feb. 2000