ソフトウェア開発過程との類似性に基づく立法支援システム
A Legislation Supporting System Based on the Analogy to Software Development Processes

角田篤泰[*1] Tokuyasu KAKUTA
齋藤大地[*2] Daichi SAITO
外山勝彦[*3] Katsuhiko TOYAMA

[*1] 名古屋大学大学院法学研究科 Graduate School of Law Nagoya University
[*2] ㈱クレステック CRESTEC Inc.
[*3] 名古屋大学大学院情報科学研究科 Graduate School of Information Science Nagoya University

This paper proposes a new method and a system to compose laws based on the analogy to software development. Though making laws has been considered an intelligent work so far, our approach can make such a work better. The essential points of our approach are 1) using legislation templates in frame-like representation, 2) focusing the patterns of the templates being limited and 3) providing a web platform for making laws in a similar way to software development environments.

1. はじめに

本研究の目的は，法令作成の過程，すなわち立法過程の作業がソフトウェア開発過程の作業と類似している点に着目し，ソフトウェア開発環境と類似した法令作成支援環境を構築し，それを法令作成の実務や教育に活かすことである．本稿ではその方法論の枠組みと試作システムについて報告する．この試作システムは eLe(e-Legislation Environment)と呼び，立法作業の実務環境と「法令コンパイラ」を提供する．法令コンパイラとは，法政策案を入力として，法令(条文集合)テキストを出力するものである．なお，このコンパイラは現在，設計中である．

2. 背景

本研究の背景については，2 つの観点，すなわち，社会的要請の観点と法律人工知能の観点から記す．

2.1 社会的要請の観点

本提案システムの社会的な必要性，とりわけ，実務の現場を考慮した必要性は次の通りである．まず，条例を作成する場合，議会に提案する条例案は，法律の専門家ではない地方公共団体の職員が作成しなければならない．地方財政が困窮している現在，職員を増やすことも専門家を雇うことも容易でないため，作業コストを抑える必要がある．また，法律を作成する中央省庁も行政のスリム化が望まれている以上，もちろんコスト削減の必要がある．次に，法令作成の作業は，記述のためのマニュアル([石毛 00]など)も多岐・詳細であり，職人的な技能も必要であり，そもそも法令なので，間違いが許されない．そのためにも作業の機械化が望まれる．最後に，本来，法令は，住民の意図に基づいて公の機関が作成するはずだが，技術的な困難さから，やむを得ず，民間業者に法令記述の作業を委託しているケースもある．やはり，「自治」というからには，自力で作成することを目指す必要があり，そのためにも本提案システムは有効である．

2.2 法律人工知能の観点

法律人工知能分野では，事実に法律を適用して，法的な結論を推論する研究や，法廷内外での紛争における論争過程に関する研究が中心である[角田 02]．このような従来研究と本研究との比較を表 1 に示す．

表 1　従来研究との比較

	従来研究	本 研 究
対象過程	司法	立法
主なユーザ	法曹	行政職員
対象過程の目的	判決	政策実行
主な機能	推論	法令作成

また，獲得された知識の利用方法も，裁判や法的なトラブルを想定した法律エキスパートシステムで利用される場合とは異なり，予防法務の場面や素人が法律の内容を問う場面で利用される注釈システムのような利用を想定している．

さらに，法令に着目して，従来研究のアプローチと本研究を比較するとき，喩えて言えば，従来研究は法令を入力とする「インタプリタ」や「逆コンパイラ」の提供であり，本研究は法令を出力する「コンパイラ」の提供を目指している．すなわち，政策案という仕様や要綱というソースから法令というオブジェクトコードを作成するコンパイル過程が立法過程であり，本研究の対象である．これに対して，従来は，まず法令を前提にして，解釈・実行をシミュレートすることや，法令や事実の知識から，後に解説する「解釈ルール」と呼ばれるものや概念階層などの知識を抽出することを対象としていたのである[Kakuta99]．

このような従来のアプローチも，多くの法律専門家が法解釈や司法に関する専門家であることから，当然の要請であり，重要な研究ではある．しかしながら，コンパイラの前に逆コンパイラを作成することは，やはり奇妙な印象を与えるであろう．

また，本研究の成果は従来研究の不足部分を補完する関係にもある．法令はいわば，「制約ルール」の集合であり，「生成ルール」をほとんど含まない．そこで，法令だけで演繹推論を行おうとすると，integrity constraint で演繹推論を行うような，根本的におかしな状況に陥りがちである．そこで，実際には「解釈ルール」というものを導入したり，これを自動取得したりするという方法で対応されてきた．これらの方式の場合，その本質は，事例ベース推論や類推，あるいはアブダクションを用いたものであり，演繹ではない．演繹にこだわっていた理由の一つは裁判の判決を導くような推論をイメージしていたからであろう．日本などの成文法の国家では，裁判官は，事実(A')に法令(A→B)を適用して，判決(B')を示す必要がある．これは「法的三段論法」などとも呼ばれる．たしかに，これ自体は演繹の形であるが，実際には，A'と A を結ぶためのルールが必要になる．これが「解釈ルール」である．しかし，これらは法令には一切書かれていないた

連絡先：角田篤泰，〒464-8601 名古屋市千種区不老町 名古屋大学大学院法学研究科，kaku@law.nagoya-u.ac.jp

め，結局これを外から入力するか，自動的に獲得する仕組みがないと，法的三段論法の演繹形に持ち込むことができない．そこで，従来のアプローチでは，演繹以外の推論の占める割合が高いのである．結局，研究の関心は解釈ルールの獲得に向かうことになる．筆者も従来はそのようなアプローチを採っており，学説や立法時の意図などの背景知識を事前に与えておき，問題に応じて切り替わる概念階層とその解釈ルールを内部的に自動生成させていた[Kakuta99]．

しかし，その所与の知識が学説の場合，学問的に面白いものや基本的な法律に関するものに限られ，日常的に人や企業が直面する圧倒的多数の行政法規については，条文ごとの細かな学説が存在しない．一方，立法時の意図は，必ず存在したはずであるが，これらは作成者の記憶の中や中間的なドキュメントの中に埋没している．この作成者とは中央省庁や地方公共団体の行政職員である．そこで，これらを集めればよいのではないかという発想が生まれた．ドキュメントになっているものであれば収集できそうだからである．従来研究ではここまでは行っておらず，本研究の成果によって補うことが可能となる．なお，このようなドキュメントのうち主なものは，「法政策要綱」や「法令案要綱」である．法令案要綱とは条文に対応するくらい具体的な，喩えれば「詳細仕様書」である．法政策要綱とは，より抽象的な「内部・外部仕様書」や「基本設計書」に対応するものである．

しかしながら，これらドキュメントをソースとする遡及入力は，作業コストが高くなり，現在の研究段階では実現が厳しい．もちろん人々の記憶にしか存在しない場合は一層困難である．そこで，当面は過去の法令についての遡及入力よりも，将来へ向けて，立法時の意図を反映した法政策の知識獲得を目指した．本研究で提案するように，まず，立法時の一連の作業を統合システム上で行うことにより，内部的には，法令の意図に関する情報を自動的に蓄えておく．これらを蓄積して，政策パタンを獲得すれば，逆コンパイラの仕様が確定し，その開発が可能になる．これによって，遡及入力しなくても，ある程度は，このような知識を自動的に再合成できるようになる．仮に人手で行う必要があったとしても，この前処理によって，大幅に負担が軽減するだろう．

こうして，実務支援や従来研究の補完のために知識を蓄えるだけでなく，逆コンパイラを作成するためにも，まずは，本研究のようなアプローチを採り，法令作成支援環境や法令コンパイラを提供することが有効であると考えたのである．

3. 立法過程とソフトウェア開発過程の類似性

一般的な立法過程の各ステップは順に，①課題設定，②基本設計，③詳細設計，④審議・決定と進行する [礒崎 05]．本研究の対象である③までをより詳細に示すと，次の通り [大島 04]．

①立案の動機となるニーズの把握，立法事実の収集・整理，政策・対策の形成
②法政策の選択・形成，関連法規の検討・調査，法令案骨子の形成(法政策要綱の作成)
③法令案要綱の構成，条文化

これをソフトウェア開発に対応付けると，次の通りである．

①調査・要求定義
②外部設計・内部設計
③詳細設計・コーディング

両者の全体的な対応関係を図 1 に示す．もちろん，この図のように④についても，審議・決定とテストが対応しているとも言えるし，法案はリロケータブルな「オブジェクトコード」であり，議会の承認を経て，具体的に施行日などが決まり，現実社会に「リリース」されたり，「ロード」されたりして効力を持つとも言える．

それでは，このような類似性が認められるとして，ソフトウェア開発手法のような考え方やフレームワークがどこまで移植できそうかという点について述べる．

まず，法令作成とソフトウェア開発の本質的類似点の一つは，その物理的出力が，いずれも，ある文法に従ったテキストであることである．特に日本の法令については，法令記述の厳密なマニュアル[石毛 00]が存在しており，プログラミング言語や機械語のように，素人を苦しめるには十分の文法や慣例が存在している．そして，法令案要綱など，法令の形式にされる直前の文書は，法令の文法に比べれば素人にとって，可読性の高い自然言語で書かれたものであり，この記述過程を高級言語によるプログラミングと見立て，さらに，それをコンパイルして法制執務の文法に従った法令を出力すると考えれば，この処理はソフトウェア開発とあまり変わらない．違いは，現在，法令案要綱は人工言語で書かれていないので，プログラミング言語のような処理しやすい書式があるわけではない点である．そこで，この部分の形式的記述方法を定義すれば，法令の作成環境はソフトウェアの開発環境と似たものを流用できる．将来的には UML のような上流工程の言語を扱うツールを導入できる途も開けるだろう．

次に，分析を進める中で，ソフトウェア開発過程におけるオブジェクト指向風のクラス定義やその継承による設計のように，法令作成の過程でもオブジェクト指向風に設計できることが分かってきた．つまり，法政策をクラスとして定義したり，既存クラスの継承や修正によって設計したりできるのである．もちろん，例外もあり，さらなる詳細な分析が必要な法政策も多数存在するが，条例などに現れる典型的な法政策に関しては，この設計が可能である．ただし，この立法過程で設計されるものは，政策によって実現される現実社会そのものをモデリングしているわけではない点に注意を要する．そうではなくて，法政策で提示される法令構造のモデリングがオブジェクト指向風に可能であるということである．

4. 法政策パタンに着目した立法支援方式

本研究で提案する方式は，簡単に言えば，法政策にはいくつかのパタンがあることに着目して，そのパタンに沿って法令作成を進めるというものである．典型的な法政策のパタンには「許認可」「資格付与(免許制度等)」「届出・登録」「経済的誘導(補助金給付や免税等)」があり，さらに，人々の行為や事業に対して単純に禁止や命令を行うだけのようなものもある．また，どの法政策にも大抵付随している規程のパタンとしては，「罰則」や「雑則的事項(不服申し立て，継承，委任等)」がある．これらの典型例の場合は，さらに，それぞれに特徴的で典型的な制度の詳細部分が組み込まれており，たとえば，「許認可」のカテゴリの制度であれば，「許可基準」や「監督」に関する制度が付随しており，「資格付与」であれば，「試験」制度などが付随している場合が多い．これら制度の骨格に関しては，従来制度とさほど変わらないものが多く，実際に，自治体の条例などでは，他の自治体の条例を土台にして設計を進める場合もある．

すなわち，エンドユーザがゼロから法政策のオブジェクトを作成する必要性は比較的少ない．仮にその必要があったとしても，基本法のような制度を作る場合であり，例外的と考えてよい．そこで，多くの場合は，我々システム提供者側から法政策のパタンに対応するクラスを提供しておき，ユーザは，提供されたクラスライブラリの中から法政策のクラスを選択できるようにした．システム上では，選択したクラスに応じた作成画面が現れるので，

その画面で値を書き込んだり，属性追加を行ったりするだけで，政策設計を進めることができるようにする．このようにすることによって，エンドユーザの負担は大きく軽減される．

なお，本研究で提供されるシステムは，統合環境として提供されるものなので，制度設計時の様々な作業に役立つような機能も，オールインワンで含まれているように設計されている．

このようにして構築された法政策オブジェクトは，必要に応じて自然言語や XML による要綱として出力されたり，コンパイルされて法令案として出力されたりする．本研究はこのような方式を提案する．

5. 立法過程支援システム eLe の試作

5.1 システムの概要

今回試作した立法過程の作業支援システム eLe について概要を記す．

本システムは，Eclipse のような，ソフトウェア開発環境としてポピュラーな IDE 系のインタフェースと同様に，ユーザがプロジェクトを作成して，そこに一連の政策を作り込んでいくという方式を採用している．大きく 3 画面からなり，上部がメニューやシステムのラベル表示部であり，下部左側がナビゲートビュー，下右側がワークスペースである(図 2 参照)．

メニューでプロジェクトビューを選ぶと，ナビゲートビューがプロジェクトをルートとする画面に切り替わる．このプロジェクトは，「立法事実」「ステークホルダ」「政策パッケージ」「作業」などの属性に割り当てられたフォルダ・オブジェクトを持っている．「立法事実」とは法政策を実施するために，実際に要望や問題が生じていることを示す事実である．システム上では，その事実を示す証拠資料や関連データが蓄積される．「ステークホルダ」とはその政策が施行されることによる利害関係者である．住民や国民一般の他に，直接に影響を受ける団体や政治家などもこのフォルダに登録される．現在は未実装であるが，たとえば意思決定支援ツールと連動させるためのデータや各関係者に関する情報を蓄えておく．「政策パッケージ」のフォルダには，まさに設計中の法政策のオブジェクトが登録される．「作業」フォルダは，立法過程の一連の作業をプロジェクトと見立てたときの各タスクを登録するものである．このフォルダのビューはデフォルトではガントチャート形式になっている．

本システムの中心となるのは，政策パッケージであり，ここを展開すると，一般的にはいくつかの関連政策が並ぶ．通常は何らかの法政策オブジェクトを 1 つ入れておく．これが，実際に法令へと変換されていくソースとなるものである．

ユーザは，法政策を選択する時に，通常はゼロから作成するわけではなく，いくつかのテンプレートとなるクラスの中から適当なものを選択し，それをベースに法政策要綱としての法令の仕様を組み立てていく．

ユーザが法政策のオブジェクトをクリックすると，その構造に応じたツリーがナビゲート画面に表示される．各ノードとなるオブジェクトを選択すると，そのオブジェクトの構築に適した作業用のビューが右のワークスペースに現れる．このようにして，そのノードに対応した設定や子ノードの付加を行っていくことで設計を進める．このツリー構造はクラスの内部構造に対応している．

これらが終わると，実は概念的には法政策要綱を構築したことになり，内部的にその情報を持っていることになる．従って，法政策要綱として可読性の高い自然言語で表示することも，より詳細な法令案要綱として自然言語や XML で表示することも可能となる．

最後に，メニュー画面の「コンパイル」をクリックすると，コンパイルが始まり，その出力として，ワークスペースには法令が表示される．

5.2 法政策の知識構造

本システムで用いられる法政策知識構造の一部を図 3 に示す．この構造は日々分析と更新を進めているが，詳細を記すためには多くの紙面が必要になるうえ，AI 技術の話から逸れてしまうので，筆者らの所属部局の紀要等での紹介を予定している．

5.3 実現

本システムは Zope2.7.5-final[*1] を用いて構築した Web アプリケーションである．HTML と Zope の基本機能だけで実現しており，JavaScript は用いていない．また，Zope のオブジェクト指向風のデータベースをそのまま利用しているため，他の RDB サーバ等は利用していない．Web 上に実現した理由は，特に使用環境を整えなくても，すぐに市町村役場や中央省庁で利用してもらえると期待できるからである．なお，本システム上のオブジェクトはすべて URI が付されることになるので，セマンティックウェブとの親和性は高い．また，Zope の組み込み機能をそのまま使って，法政策オブジェクトの状態をすべて XML 形式で出力することも可能である．

現在，統合環境のインタフェースは開発をほぼ完了しているが，コンパイラを始め，自然言語出力の部分は未実装である．また，クラスライブラリは，法政策トップクラスと許可制度のクラスのみ概ね実装済みである．これらについては，日々，分析とともに蓄積している．

6. まとめ

本稿では，法令作成のプロセスをソフトウェア開発プロセスと類似の統合環境上で進め，従来の職人的作業をより客観化・効率化するための方式の概要を示した．現在も試作システムの開発や法政策の知識分析を進めているが，その過程で，法令が詳細な表記規則に従っていること，および，典型的な法政策にはパタンが存在することから，現実的なシステムの提供が可能であることも述べた．

法政策にはパタンがあると言っても，まだまだ分析は始まったばかりであり，知識の蓄積が肝要である．またその現実的な蓄積を計画することも必須である．eLe に関してはコンパイラなど形式的処理の部分について開発を進め，検証を行う必要がある．以上の要請への対応は今後の課題である．

参考文献

[石毛 00] 石毛正純: 法制執務詳解〔三訂版〕, ぎょうせい (2000).

[Kakuta99] Kakuta, T. and Haraguchi, M.: A Demonstration of a Legal Reasoning System Based on Teleological Analogies, Proc. of ICAIL99, pp.196-205 (1999)

[角田 02] 角田篤泰, 原口誠: 法的推論と類似性 ― 対話と議論の観点から, 人工知能学会誌, 17-1, pp.14-21(2002).

[礒崎 05] 礒崎初仁: 条例をつくる(2), ガバナンス, 2005-2, ぎょうせい, pp.128-129(2005).

[大島 04] 大島稔彦(編著):法令起案マニュアル, ぎょうせい (2004).

[*1] http://zope.org/

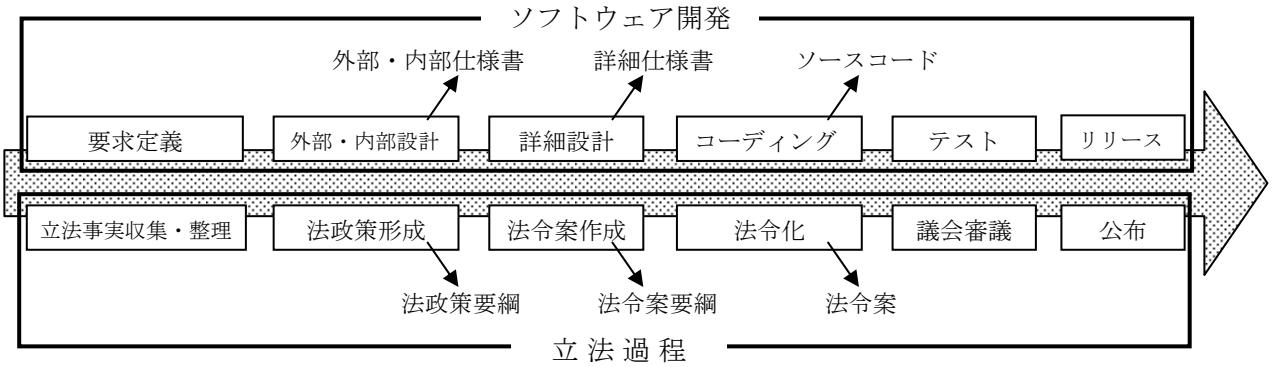

図1　立法過程とソフトウェア開発過程の類似性

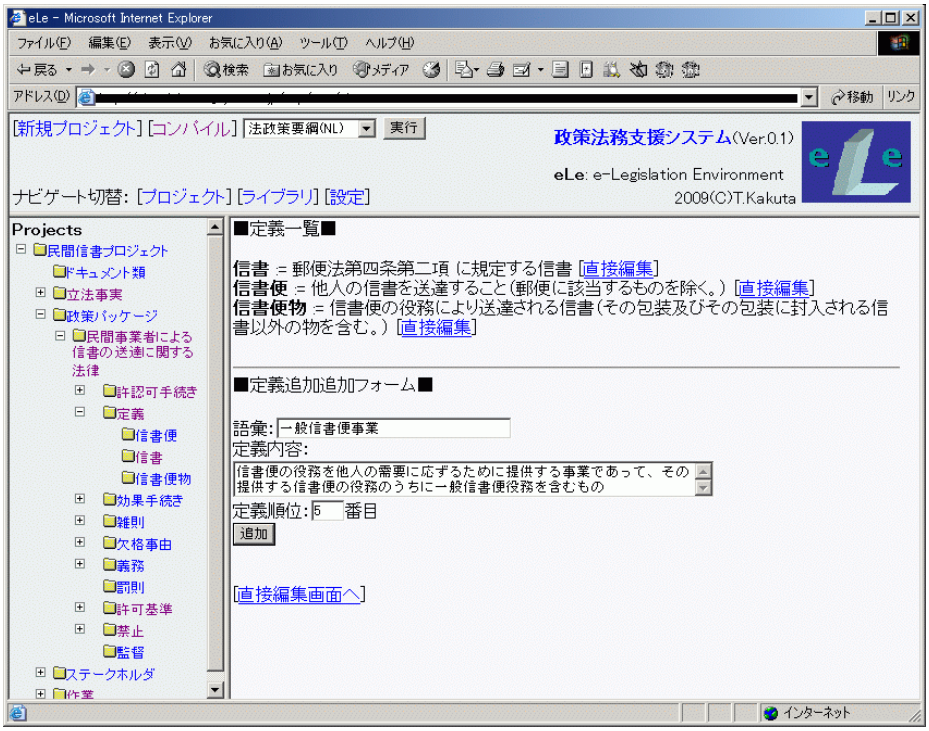

図 2　試作システム eLe の画面例

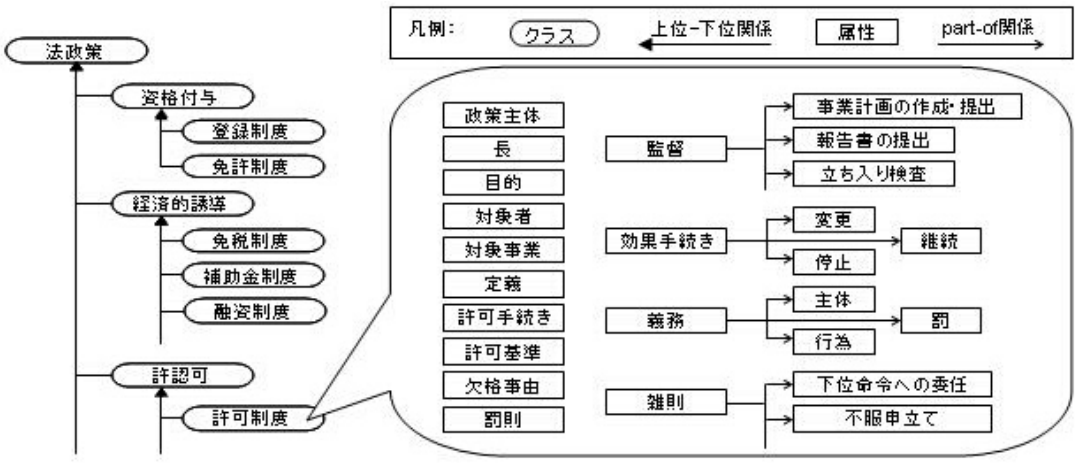

図 3　法政策知識の構造(一部)