Vol.2014-SE-186 No.7
2014/11/13

スプレッドシートを利用したビジネスルールエンジンの実装

小池賢一[†1]　大松史生[†1]　谷屋直隆[†2]　小清水 麻衣[†2]

スプレッドシートを使用したビジネスルールエンジンを提案する．従来は，スプレッドシートに自由にルールを記述することができず，ルールの内容がディシジョンテーブルなどに限られていた．本研究では，スプレッドシート上でルールを評価することにより，複雑なルールを記述できるようにする方式を提案し実装例を示す．

Implementation of Business Rule Engine using the Spreadsheet

KENICHI KOIKE[†1]　FUMIO OMATSU[†1]
NAOTAKA TANIYA[†2]　MAI KOSHIMIZU[†2]

We propose the implementation of Business Rule Engine using the spreadsheet. The spreadsheet itself is general-purpose, but the variation of rule is restricted to Decision Table etc. In this paper, we propose a method of creating a complicated rule evaluating rule on a spreadsheet. And we describe an example implementation of this method.

1. はじめに

従来から業務アプリケーションにおいて，ビジネスルールを処理するためのミドルウェアとしてビジネスルールエンジン BRE（Business Rules Engine）やビジネスルールマネージメントシステム BRMS（Business Rules Management System）が広く利用されている．ビジネスルールエンジンを利用することにより，ビジネスロジックをルールの形式で記述して，アプリケーションから外出しにすることができ，業務管理者が直接業務ルールを把握・修正することが容易になり，業務アプリケーションの開発コストと運用コストの低減が実現できる．しかし，既存の方式では専用のルール記述言語を使用してルールを記述するため，新たに言語を習得す必要があった．また，スプレッドシートを利用してルールを記述する場合は，ディシジョンテーブルなどの一部のルールのみに限定される制限があった．さらに，スプレッドシートに記述したルールは通常のプログラミング言語で作成したルールと同様にテストする必要あるが，ルールのどこに問題があるか確認できるようにしてテストの工数が増えないようにする必要がある．ルールの数は数千の規模になることがあるため，ルールを評価の実行時の性能の要件を満たす必要がある．

そこで，ビジネスルールをスプレッドシート上に日本語で記述する方式を提案する．ルールは，実行する段階で Java のプログラムに自動変換してから実行する．今回は，広く利用されている Microsoft Excel のファイル形式でルールを記述することを想定する．以下の第 2 節で本稿の提案するシステムの構成とスプレッドシートの記述例を示す．第 3 節で本稿の解決する課題，第 4 節で課題の解決方法，第 5 節にまとめを記述する．

2. システム構成

以下に本稿で提案するシステムの構成を示す．

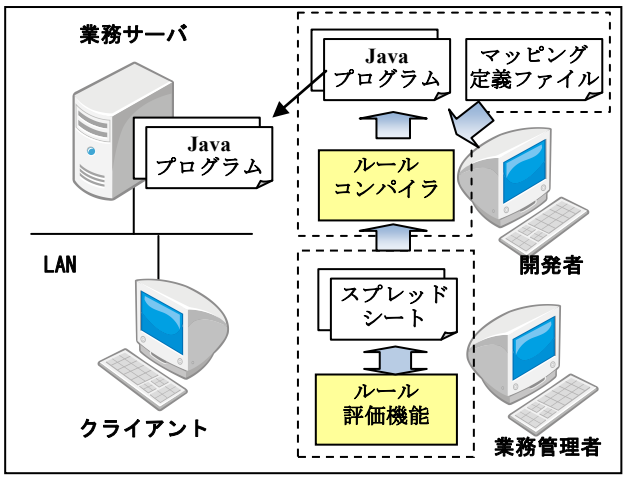

図 1　システム構成

本システムでは，スプレッドシートの作成作業は業務管理者が行い，作成したスプレッドシートはルール評価機能で正しいこと簡易的に確認する．以下にルールの例を示す．

ルール	評価結果
支払方法="カード" and 一括支払い=True	FALSE

図 2　ルールの記述例

「ルール」というタイトルの下にルールが記述されており，この例では支払方法が「カード」であり一括払いの場合に成り立つルールを表している．また，「評価結果」の下には，仮に入力した値に対して評価を実行した結果を表示

†1 三菱電機株式会社情報技術総合研究所
Information Technology R&D Center, Mitsubishi Electric Corporation
†2 三菱電機インフォメーションシステムズ株式会社
Mitsubishi Electric Information Systems Corporation

している．ルールの評価はルール評価機能によりスプレッドシート上で実行する．以下に評価で使用した仮の入力値の例を示す．

入力情報	
支払方法	カード
一括払い	FALSE

図 3　入力値の例

ルールをシステムに反映するためには，作成したスプレッドシートを開発者に渡し，開発者は，日本語で記述されたルールの日本語と業務アプリで使用している変数名を対応づけたマッピング定義ファイルを作成する．以下にマッピング定義ファイルの例を示す．

```
<?xml version='1.0' ?>
<mapping xmlns="http://xxx.co.jp">
  <var name="pay" type="text" excel="支払方法" />
  <var name="one" type="boolean" excel="一括払い" />
</mapping >
```

図 4　支払方法

この例では日本語の「支払い方法」と変数「pay」が対応しており，「一括払い」と「one」が対応していることを示している．

開発者は，作成したマッピング定義ファイルとルールを記述したスプレッドシートを，ルールコンパイラに渡して，Java のプログラムに変換する．生成した Java のプログラムは，JUnit などのテストフレームワークでテストを行い，テストに合格した場合は，コンパイルして業務サーバに格納する．以下にスプレッドシートに記述できるルールの種類を以下に示す．

表 1 ルールの種類一覧

No	ルール種類	説明
1	基本ルール	論理値や変数，関数，括弧で囲まれたルールなどを表す
2	否定ルール	値が真偽値になるルールの否定
3	and ルール	値が真偽値になるルールの and 条件を定義
4	or ルール	値が真偽値になるルールの or 条件を定義
5	条件分岐ルール	Excel の IF 文を一つ以上記述し，上から評価して条件が成立する最初の IF 文値を使用する

3. 課題

3.1 ルールのテスト方法

ルールコンパイラによって生成された Java プログラムは，JUnit などのテストフレームワークを使用して単体テストを行う必要がある．テストの結果誤りが検出された場合は，スプレッドシートを修正した後，再度ルールコンパイラを使用して Java を生成する必要がある．そのため，スペルミスのようなケアレスミスの場合でもこのサイクルを繰り返すことになり，開発効率が低下する原因になる．

3.2 ルール評価時の課題

業務システムで定義されるルールの種類は，目的ごとに分けることができる．以下にルールの用途の具体例を示す．

(1) 入力値の書式チェック用ルール
(2) 入力情報の不整合チェック用のルール
(3) 入力制限チェック用のルール
(4) 他システムとの連携時のチェックルール
(5) 処理結果確認用のチェックルール

業務システムの規模により，数千規模のルールを作成することがある．また，毎秒 10 件のトランザクションが発生するシステムでは，数千のルールの評価を 0.1 秒のオーダーで結果を返すことが求められる．

ルールを評価する方法として，単純に各ルールを一つずつ評価する方式が考えられる．しかし，この方法では，各ルールの項目が数十あるような複雑なルールの場合は，0.1 秒以内で処理を行うことができない．一方，プロダクションシステムでは 1974 年に提案された Rete アルゴリズムが使用されている．Rete アルゴリズムでは，ルールをノードのネットワークに変換して，ノードを辿ることでルールの判定を行う．

この方式の問題はネットワークの構築や修正の処理が複雑であるため，実装するために工数が掛かるという問題がある．

ルール評価時の課題を以下に記述する．

A) ルールの評価を高速に実行する
B) ルールの評価処理が複雑になる

4. 解決策

4.1 ルールのテスト方法

スプレッド上でルールを評価する「ルール評価機能」を開発することで，スペルミスなどのケアレスミスの場合は，ルールをコンパイルしないでスプレッドシート上で確認できるようにする．以下に，スプレッドシート上で「ルール評価機能」でルールを評価した結果を示す．

ルール	評価結果	個別評価結果
支払方法="カード" and 一括支払い=True	FALSE	TRUE FALSE

図 5　ルールの例

「個別評価結果」の欄には，ルールを構成する個別のルールの評価結果を示しており，ここでは二つの値「TRUE FALSE」を表示している．左の TRUE は「支払方法="カード"」の評価結果を示しており，右の FALSE は「一括払い=True」の評価結果を示している．このように，ルールを構

成する個別のルールの評価結果を示すことで，業務管理者は自分が正しいルールを記述できているかを確認しながら作業を行うことができる．さらに，エラーが発生した場合は，個別評価結果を参照することで，ルールのどこでエラーが発生しているかを把握することができる．以下にエラーが発生した場合の例を示す．

ルール	評価結果	個別評価結果
支払方="カード" and 一括支払い=True	#NAME?	#NAME? FALSE

図 6　エラーが発生した場合例

この例では，左のルールの「支払方法」の項目名に間違いがあり「支払方」と記述しているために，ルール評価結果が「#NAME?」というエラーコードになっている．業務管理者は，「支払方="カード"」でエラーが発生していることを把握して，「支払方」を「支払方法」に修正する．エラーコードは Microsoft Excel と同様のエラーコードを出力する．以下のエラーコードに対応している．

表 2 エラーコードの種類一覧

No	エラーコード	説明
1	#NAME?	変数名や関数名が間違っている
2	#NUM!	数値が大きすぎる場合と小さすぎる場合
3	#VALUE!	引数の種類が正しくない
4	#DIV/0!	割り算の分母がゼロ

スプレッドシート上でエラーが表示されなくなった段階で，そのスプレッドシートを開発者に渡す．開発者は，スプレッドシートをルールコンパイラで Java にコンパイルするが，スプレッドシートでケアレスミスは修正されているため，コンパイル時のエラーの多くは，修正済みであり，そのような理由でコンパイルを繰り返すことを避けることができる．生成された Java のクラスは JUnit などをテストフレームワークを使用して厳密なテストを行う．このときのテストは通常の Java の単体テストを同じ基準で実施する必要がある．

4.2 ルール評価時の課題の解決策

実際のルールでは，多数の項目を使用してルールが記述されていることが多い．以下に 4 つの項目からなるルールの例を示す．

ルール	評価結果	個別評価結果
支払方法　="カード"　and カード種別 ="SMART"　and 配達先　="自宅"　and 一括支払い =True	FALSE	TRUE TRUE TRUE FALSE

図 7　項目が多いルールの例

このルールを単純に Java のプログラムに変換すると if 文の記述が長くなり，各項目について毎回一致判定をすることになり，処理の負荷が高くなる．以下に上記のルールを Java に変換した例を示す．

```
boolean rule001(String pay, String card, String
addr, boolean one_pay) {
  if(pay.equals("カード") &&
     card.equals("SMART") &&
     addr.equals("自宅")){
     …
```

図 8　ルールの Java への変換例

そこで，ハッシュ表を利用することで文字列の一致判定を，まとめて実行する方式を採用した．上記の場合は「支払方法」と「カード種別」と「配達先」は一つの文字列にアペンドすると「カード SMART 自宅」となり，この文字列をハッシュのキーとしてハッシュ表を作成することで一度に判定することができる．ハッシュ表はクラスの初期化処理で作成しておき，ルールの実行時にハッシュに該当する値が格納されているかを確認する．以下に例を示す．

```
static Map map = new HashMap();
static private void init(){
  map.put("カードSMART自宅","True")
}
rule001(String pay, String card, String addr,
boolean one_pay) {
  key = pay+card+addr;
  if(map.get(key) != null){
     …
}
```

図 9　ルールの一部をハッシュ化した例

ハッシュ表の作成処理は Java の組み込みのクラスを利用できることから実装は単純にできる．なお，ハッシュの値としてルールの ID を格納することで，該当するルールの判定を一度ハッシュを引くだけで判定できるようになり，ルールの判定を高速に行うことができる．

5. おわりに

本稿では，スプレッドシートを利用したビジネスルールエンジンの実装方法について論じた．本方式の特長は，スプレッドシート上でルールを評価することで，ルール作成時のケアレスミスを防ぐことを可能とし，さらに，ハッシュ表で複数の項目を同時に判定してルールを絞り込むことで，シンプルな仕組みで高速に処理できるようにした点である．今後はルールのテストの自動化を検討していく．

参考文献

1)　小池賢一, 菅野幹人: "スプレッドシートを利用した業務アプリケーションの実装", 情報処理学会 FIT 公演論文集, 2011.
2)　川口正高，塩尻綾子，浅見可津志，原田雅史，佐藤啓紀: オープン環境のシステム構築を高品質･短納期で実現する Web システム開発標準 “MIWESTA"三菱電機技報 81(7), 489-492, 2007-07