Code Generate From Business Rule

by SparxSystems Japan

ビジネスルールからのコード生成 機能ガイド

(2016/10/07 最終更新)

1. はじめに

この機能ガイドでは、Enterprise Architect Suite ビジネスモデリング版およびアルティメット版で利用可能な、ビジネスルールからのコード生成機能について、流れを説明します。

この機能を利用することで、自然言語で記述されたビジネスルールから、ある程度のソースコードの実装を自動的に生成することができます。100%のソースコードを生成することはできませんが、システムの設計開発の作業を効率化することが可能になります。

なお、このドキュメントでは、Enterprise Architect13.0 ビルド 1304 を利用しています。

2. コード生成までの流れ

ビジネスルールからのコード生成機能を利用する場合には、次のような手順で分析設計を行うことになります。

1. プロジェクトファイルの作成
2. ビジネスルール(要件)の定義
3. ルールタスク要素の定義と関連づけ
4. データ構造の定義
5. クラスの振る舞いの定義
6. ビジネスルール・データ構造・処理の関連づけ
7. ソースコードの生成
8. 自動生成の対象外の範囲について、ソースコードの記述

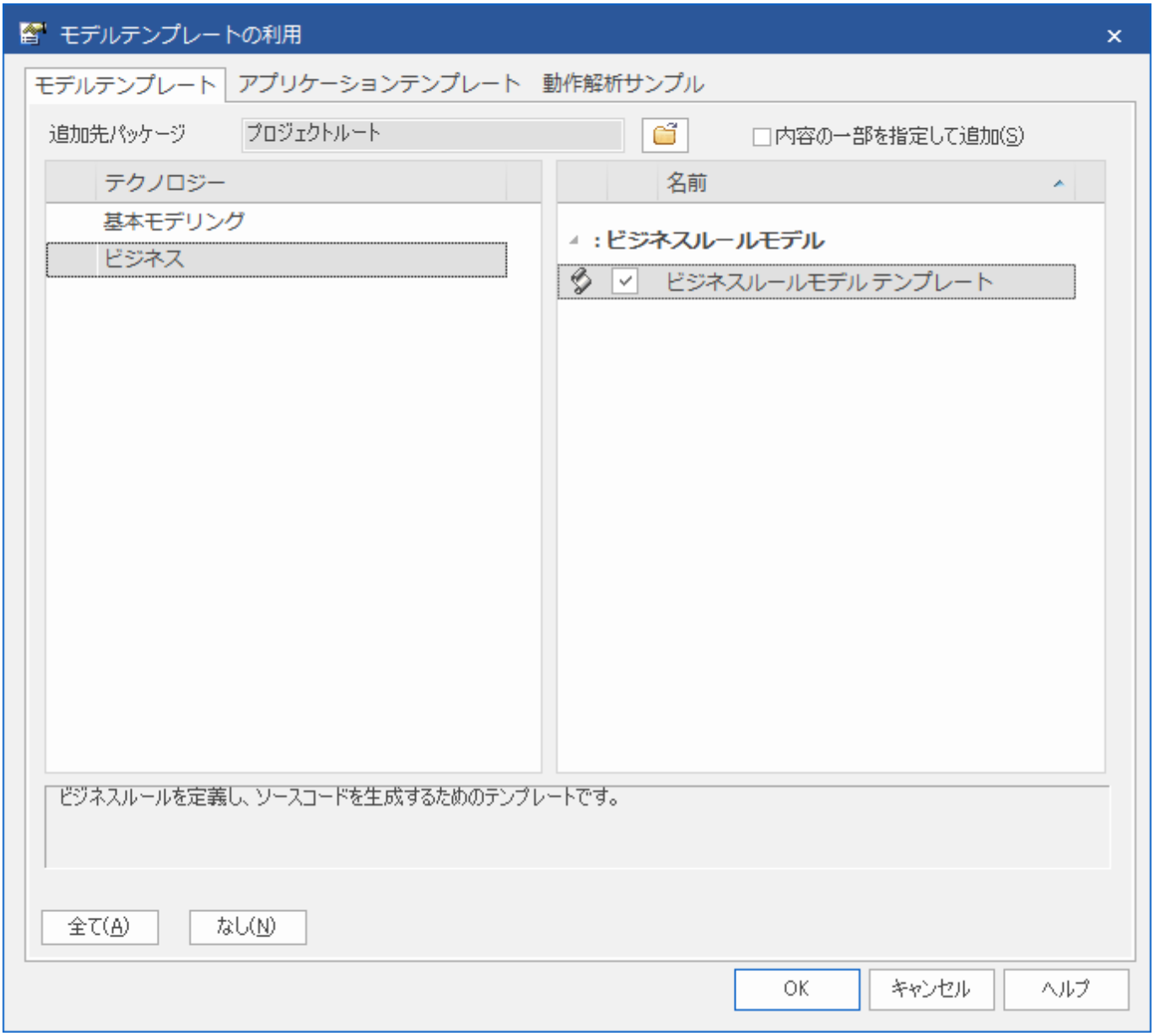
このうち、Enterprise Architect の機能として実現可能なのは 6 番までとなります。6 番までの操作を行い、その後不足する部分について、7 番で実装を追加します。

この機能で出力可能なプログラム言語は、Java, C#, C++, VB.NET です。

以下、それぞれの内容について紹介します。

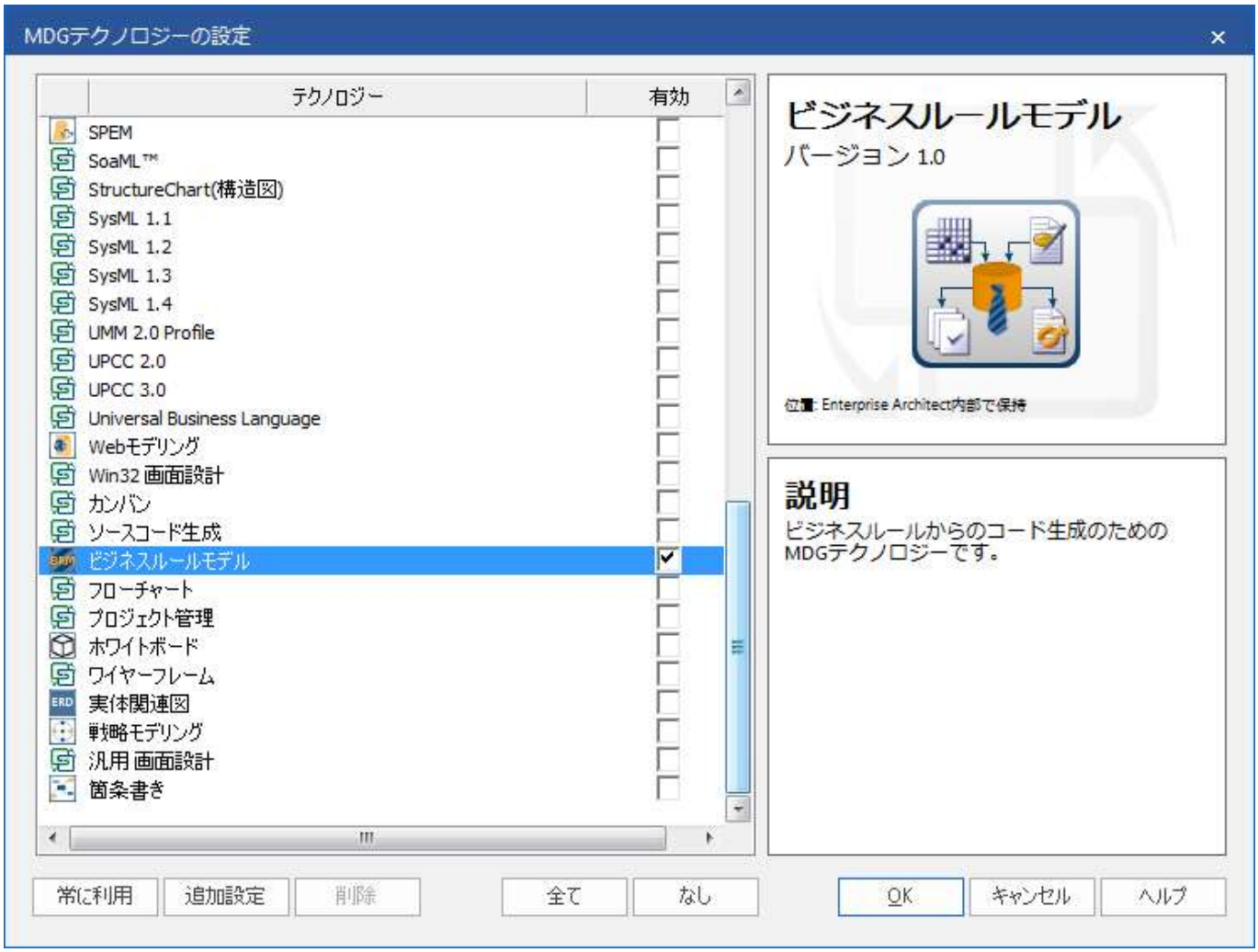
3. プロジェクトファイルの作成

まず、ビジネスルールからのコード生成を利用するための準備を行います。最初に、プロジェクトファイル(EAP ファイル)を新規に作成します。新規に作成したプロジェクトファイルには、中身が何もありません。そこで、「モデルテンプレートの利用」の機能で、「ビジネスルールモデル テンプレート」を利用します。

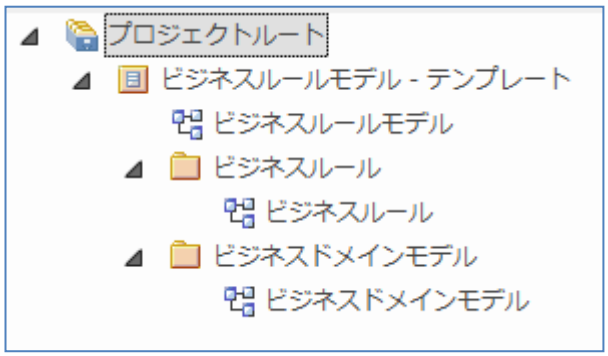


このモデルテンプレート「ビジネスルールモデル テンプレート」が表示されない場合には、MDG テクノロジーの「ビジネスルールモデル」が有効になっていない場合があります。「アドイン・拡張」リボン内にある「MDG テクノロジー」から「設定」を実行し、「ビジネスルールモデル」にチェックを入れてください。

(なお、この「ビジネスルールからのコード生成」の手順で必要な項目は、「Enterprise Architect 標準 (UML2)」および「ビジネスルールモデル」の 2 つのみです。不要であれば、他の項目についてはチェックを外してください。)



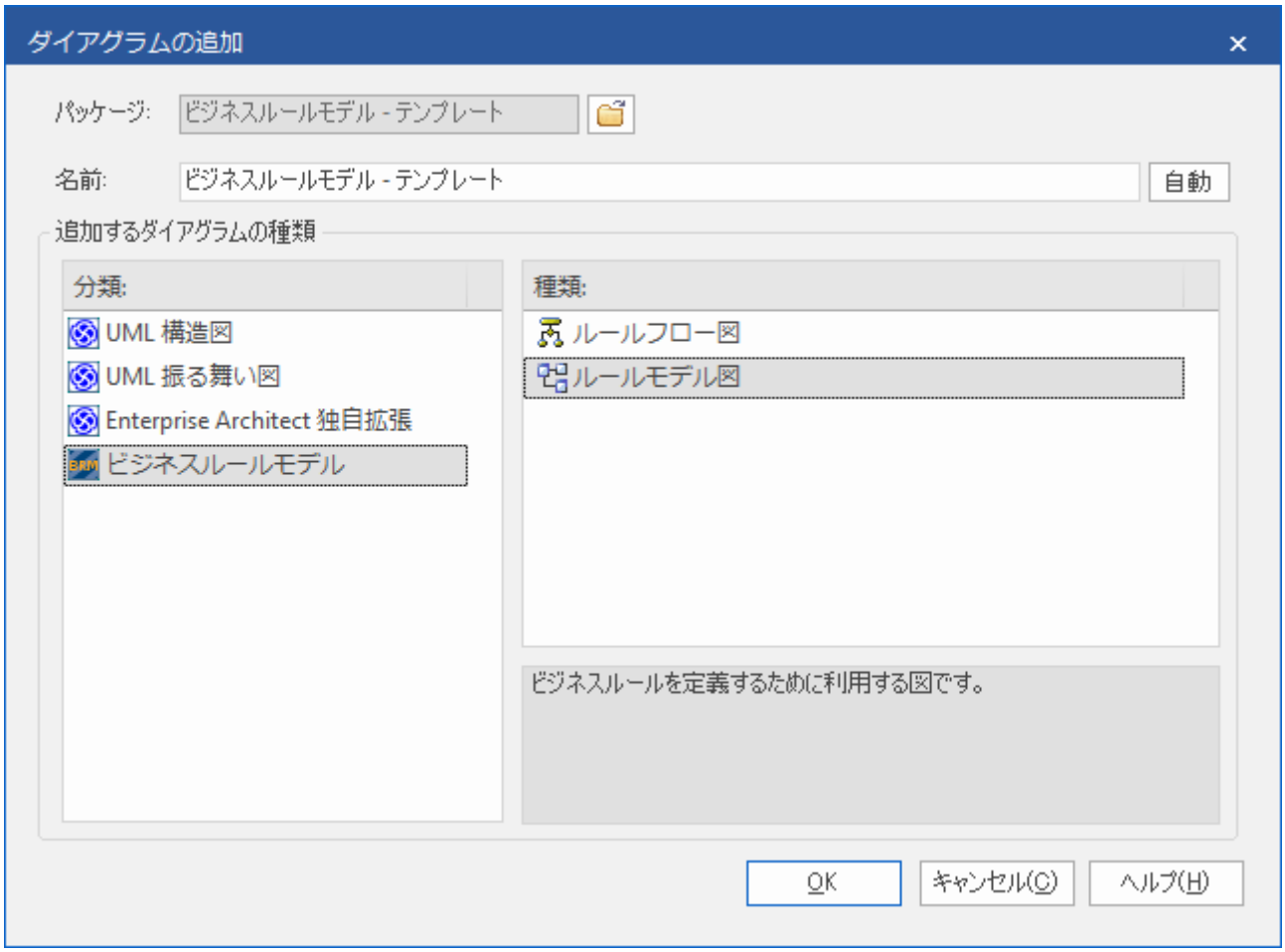
このテンプレートの内容は、次のようになっています。



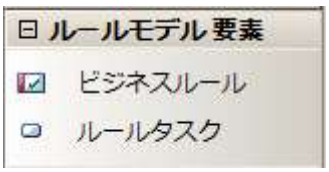
このように、テンプレートにはいくつかのパッケージとダイアグラム(図)が含まれています。それぞれの図には説明が書かれています。今回の説明では、このテンプレートから新規に設計する方法ではなく、サンプルファイル(EAExample.eap)に含まれる、レンタカーサービスのサンプルで説明します。実際にビジネスルールからのコード生成を実行する場合には、このテンプレートを参考にして内容を作成していってください。

4. ビジネスルール(要件)の定義

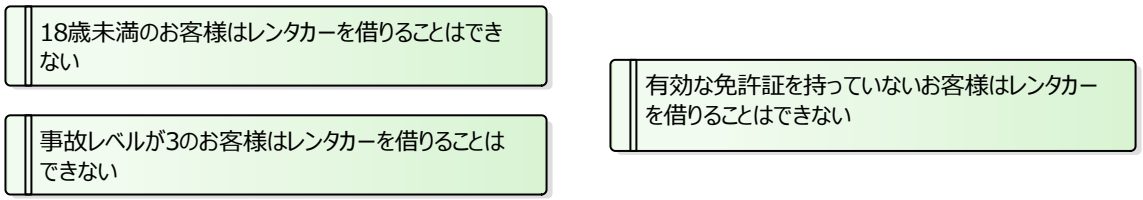
まず、ビジネスルール(要件)を定義します。この定義は、「ルールモデル図」に対して「ビジネスルール」要素を配置します。ルールモデル図を新規に作成する場合には、ダイアグラムの追加画面でルールモデル図を作成してください。



ルールモデル図を開くと、ツールボックスにビジネスルール要素とルールタスク要素が表示されます。このうち、ビジネスルール要素をダイアグラム内に配置して、ビジネスルールを定義します。



このビジネスルール要素には、日本語の文章で内容を記述します。例えば、次のように定義します。

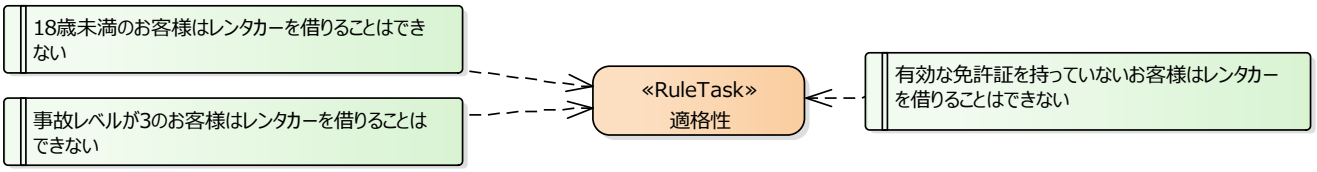


ビジネスルールでは、このような「制約」「制限」にあたるルールを定義していきます。Enterprise Architect 内で作成することもできますし、CSV 形式で Enterprise Architect に取り込むこともできます。CSV 形式で取り込む場合、Type=Requirement,Stereotype=BusinessRule として設定してください。
(手順の詳細は、ヘルプファイル「CSV の読み込み」をご覧ください。)

5. ルールタスク要素の定義と関連づけ

このようにしてビジネスルールを洗い出した後は、そのビジネスルールを分類します。具体的には、関連するビジネスルールに対して、関連性を単語で表現した「ルールタスク」要素を作成し、依存の関係で結びます。

例えば、上記の 3 つのビジネスルールを例にしますと、いずれも「借りることはできない」という制限のルールであり、レンタカーを借りることができるかどうか、という「適格性」についての判定基準になっています。そのため、「適格性」というルールタスク要素を作成しました。そして、このルールタスク要素と、ビジネスルール要素を結びつけます。(結びつけはクイックリンク機能で作成できます。)



このようにして、全てのビジネスルール要素が、少なくとも 1 つのルールタスク要素と関係が結ばれるように、ルールタスク要素を作成します。なお、1 つのビジネスルール要素が複数のルールタスク要素と関係があるような場合もあり得ます。

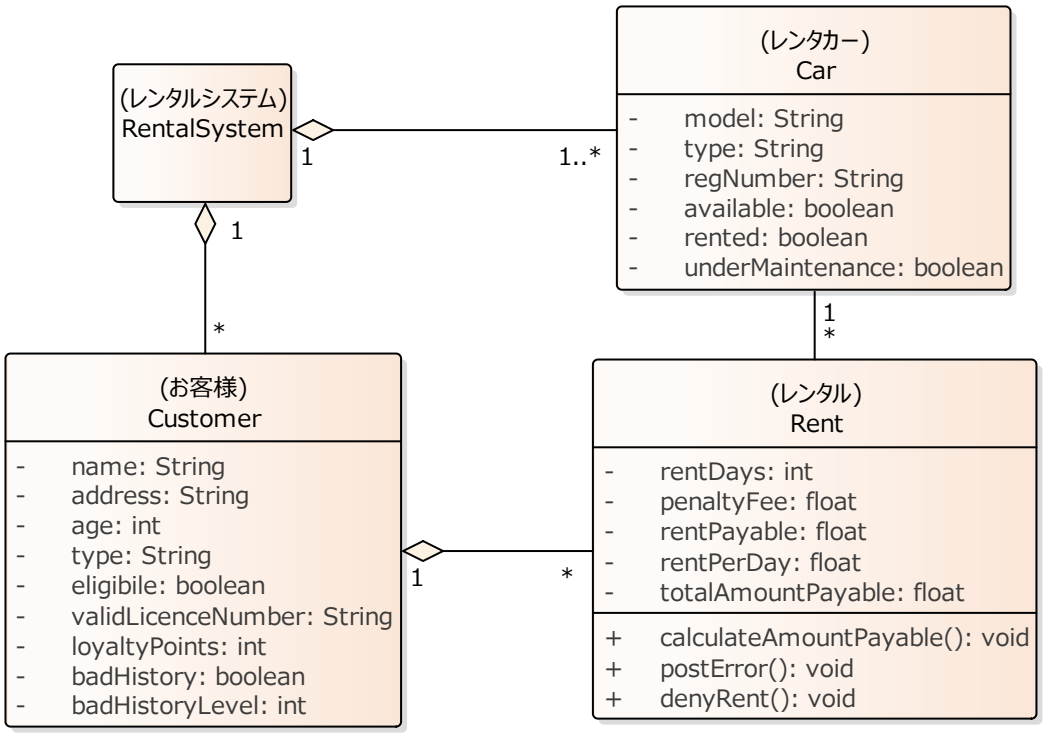
以上で、ビジネスルールに関する作業はいったん終了です。

6. データ構造の定義

次に、このビジネスルールを含むソースコードを生成するための、クラス図を作成します。このクラス図に配置されたクラスからソースコードを生成することになります。

サンプルプロジェクトに含まれる例では、次のようなクラス図になります。ビジネスルールの内容などを元に、必要と思われるデータを属性として保持するようなクラスを作成・配置してあります。なお、この段階で

は、それぞれのクラスの属性や操作が過不足なく定義されているかどうか、を気にする必要はありません。後の工程で、属性や操作の不足を確認することができます。



なお、上記のクラス図では、それぞれのクラス要素の別名に日本語を設定しています。これにより、分析段階では日本語で検討し、コード生成時にはアルファベットのクラス名で出力することが可能になります。

7. クラスの振る舞いの定義

ここまでで、ビジネスルールの定義・分類を行い、データ構造(クラス要素)をある程度決めました。次に行うことは、実際のフローの中でルールタスク要素の出現順や処理フローを定義することです。

この作業は、その処理フローを持つクラス要素に対して、ルールフローアクティビティを追加することから始まります。対象のクラス要素を右クリックして「子ダイアグラムの追加」→「ルールフロー アクティビティ」を選択します。名前を入力する画面が表示されますので、図の内容に応じた適切な名前を入力してください。

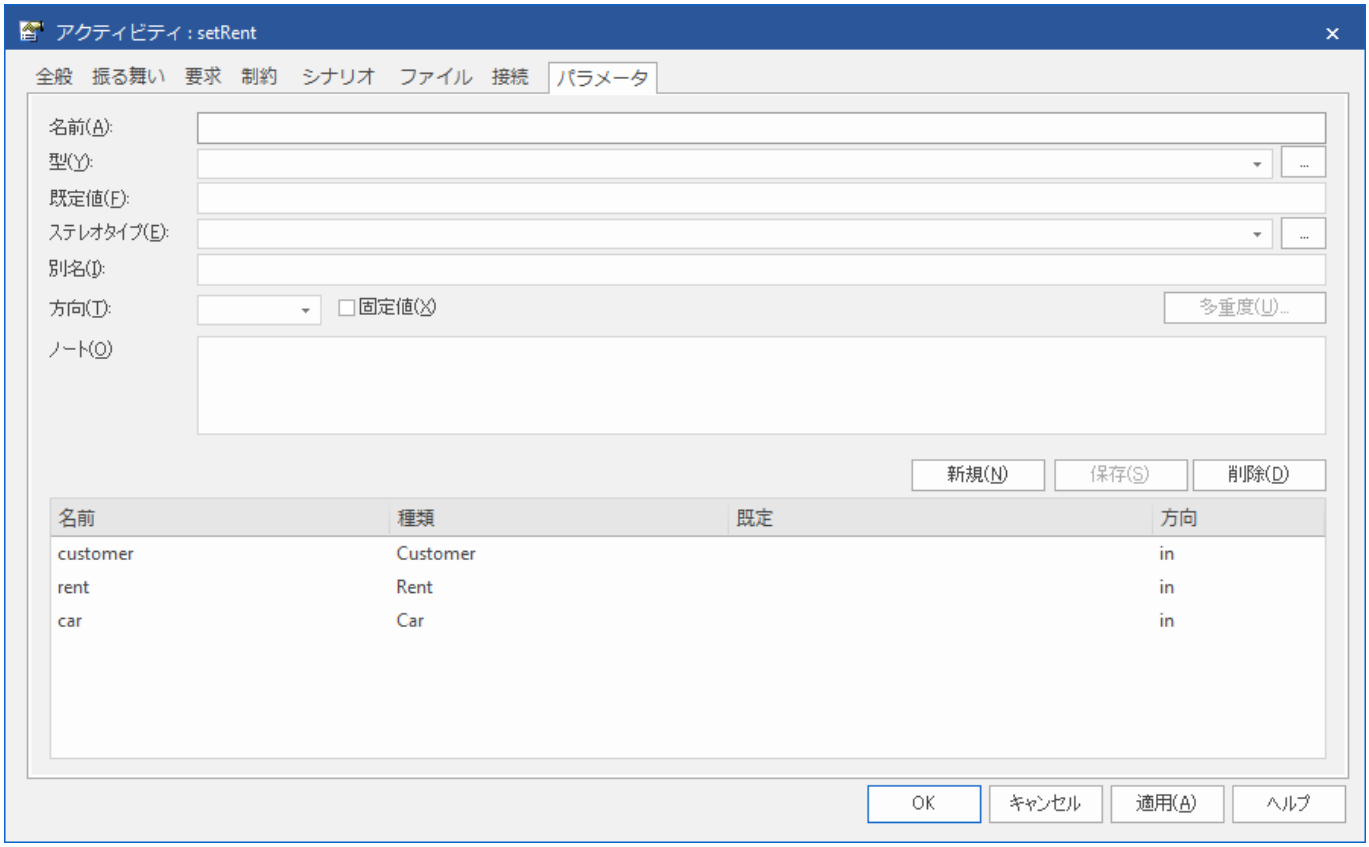
すると、対象の要素に自動的にアクティビティ要素(ルールフローアクティビティ要素)が追加され、その要素の子ダイアグラムとしてルールフロー図が自動的に追加されます。そして、その作成されたルールフロー図が表示され、編集可能になります。

このルールフロー図は、特別なアクティビティ図です。アクティビティ図と基本的には同じですが、以下の違いがあります。

- 配置できる要素は、「開始」「終了」の要素の他、分岐のための「デシジョン」「マージ」要素と、ルールタスク要素のみです。それ以外の要素は利用できません。
- 開始・終了要素は、必ず 1 つずつ必要です。開始から終了にフローが流れるように定義しなければなりません。
- フローを分岐する場合には、デシジョン要素を利用します。デシジョン要素で分岐した処理は、必ずマージ要素でフローを結合しなければなりません。
  (デシジョン要素・マージ要素を利用せず、ルールタスク要素から直接分岐することも可能です。)
- 分岐する場合、ガード条件が if 文の条件としてそのまま出力されます。そのため、if 文の内容は対象のプログラム言語にあった文法で記述する必要があります。

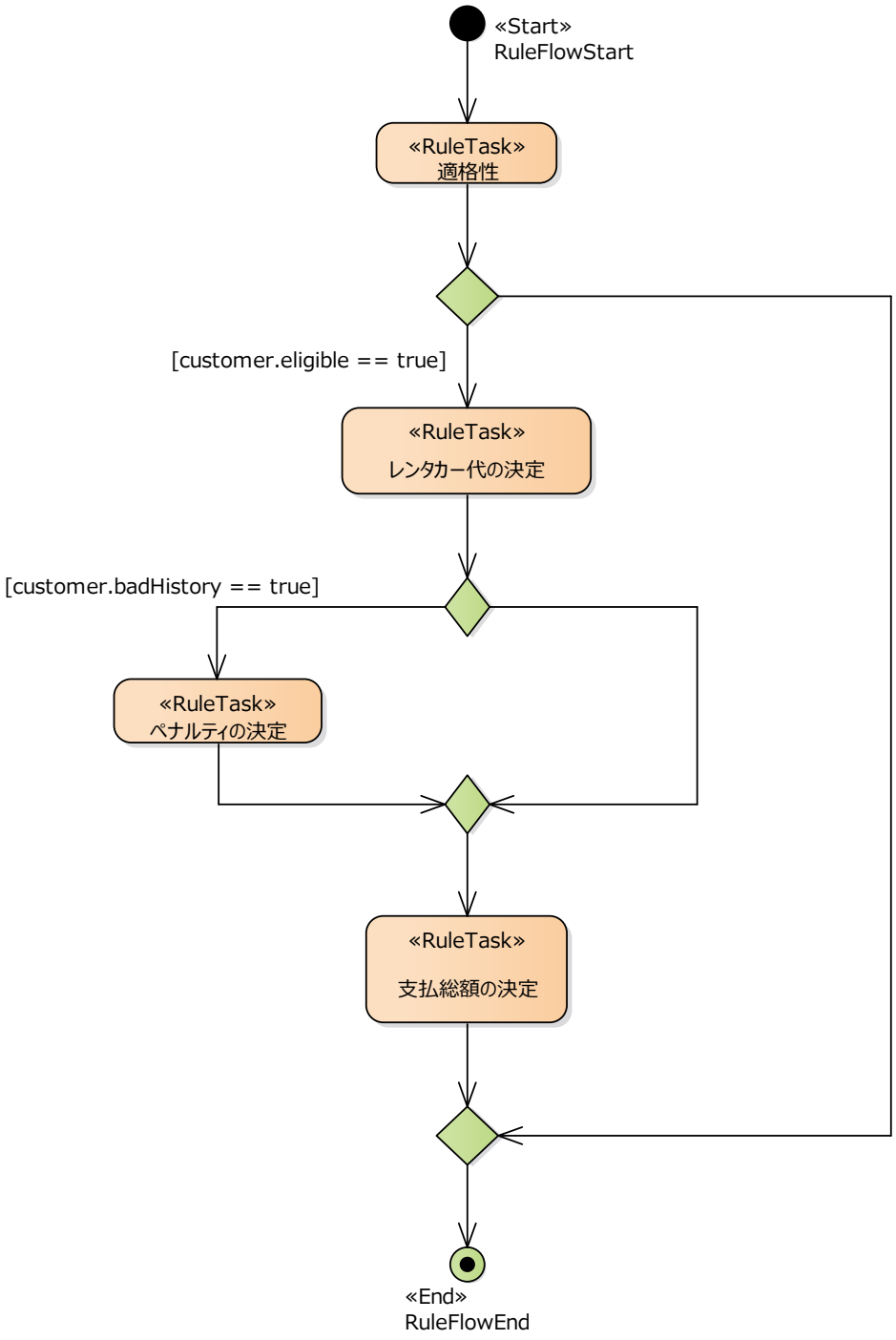
これらの内容に従って記述しないと、正しいソースコードの生成ができません。

このルールフロー図は、1 つの操作(メソッド)として出力されます。引数を定義する場合には、ルールフローアクティビティ要素のプロパティ画面の「パラメータ」グループで定義します。今回の例では、引数に対象のお客様(Customer) クラス・レンタル(Rent)クラス・レンタカー(Car)クラスを設定しました。



なお、戻り値を定義する場合、フローにおいて戻り値を返す位置に新規にルールタスク要素を作成します。プロパティ画面の「効果」グループに、「return true;」のような戻り値に関する処理を記入してください。

今回の例では、以下のようなルールフロー図になります。



作成時には、ルールタスク要素は以前作成した要素をプロジェクトブラウザ内で探し、ダイアグラム内にドロップして配置します。

8. ビジネスルール・データ構造・処理の関連づけ

ルールフロー図を定義した後は、ルールタスク要素に対して詳細なルールを定義し、今まで定義したビジネスルール・データ構造(クラス要素やその属性・操作)・処理(ルールフロー図)を関連づけます。関連づけには、「ルールコンポーザー」機能を利用します。ルールコンポーザーは、ルールタスク要素に対して右クリックし

「ルールコンポーザー」を選択することで、呼び出すことができます。

No	ビジネスルール
1	有効な免許証を持っていないお客様はレンタカーを借りることはできない
2	18歳未満のお客様はレンタカーを借りることはできない
3	事故レベルが3のお客様はレンタカーを借りることはできない

デシジョンテーブル | 計算ルールテーブル

>	ルールの関連付け		1	2	3

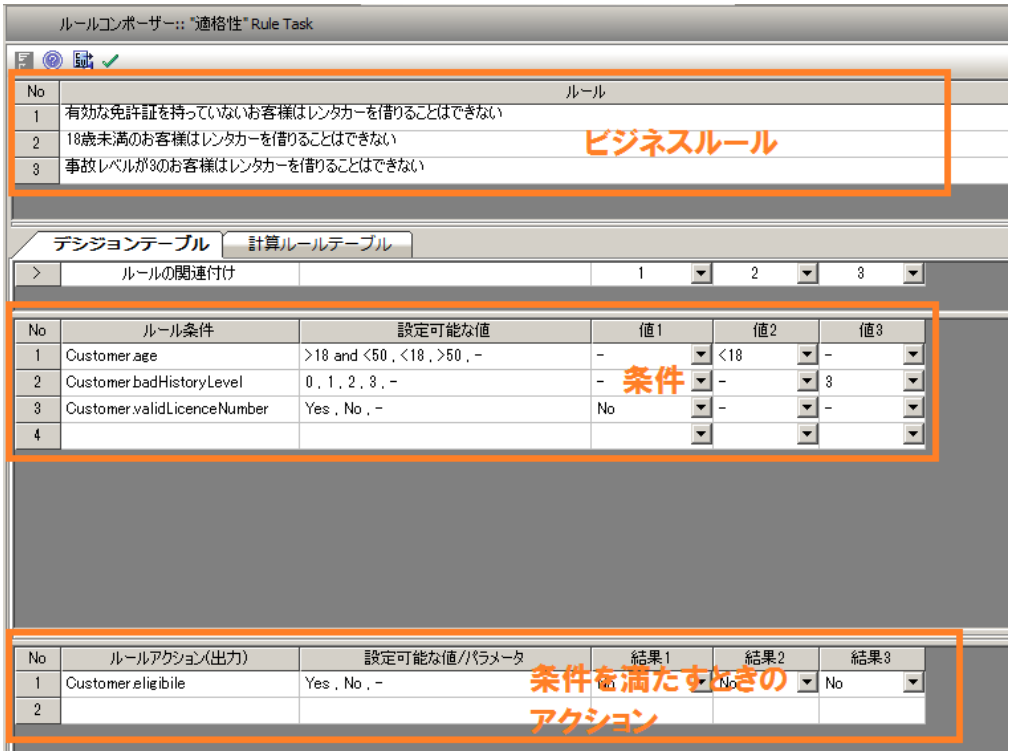
No	ルール条件	選択可能な値	値 1	値 2	値 3
1	Customer.age	>18 and <50 , <18 , >50 , -	-	<18	-
2	Customer.badHistoryLevel	0 , 1 , 2 , 3 , -	-	-	3
3	Customer.validLicenceNum...	Yes , No , -	No	-	-
4					

No	ルールアクション(出力)	設定可能な値/パラメータ	結果1	結果2	結果3
1	Customer.eligibile	Yes , No , -	No	No	No
2					

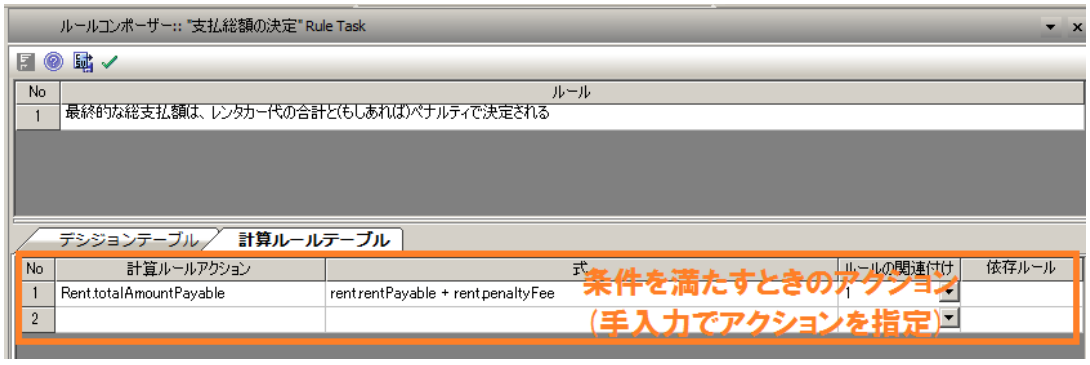
ルールコンポーザーの上部の「ビジネスルール」の表には、対象のルールタスク要素に関連づけたビジネスルール要素の名前が表示されています。このルールに対して、クラス図の属性や操作を具体的に指定することで、日本語で書かれた内容がソースコードとして出力可能な形に変換される、ということになります。

「デシジョンテーブル」タブの上部では、プロジェクトブラウザから属性や操作をドロップして、ビジネスルールの内容を示すための、具体的な条件を設定します。(操作は戻り値があるもののみ利用可能)

それぞれの条件を満たすときに実行するアクションの内容が、もし属性への値の設定や操作の呼び出しで表現できる場合には、「デシジョンテーブル」タブの下部に内容を記入します。それ以外の場合には、「計算ルールテーブル」タブに内容を記入します。



「計算ルールテーブル」タブでは、「式」として、自由に内容を入力できます。ここに入力した内容はそのままソースコードに出力されますので、対象の言語の文法に沿った内容にする必要があります。なお、特定の条件を満たす場合に「計算ルールテーブル」タブの内容を実行する、という場合には、「デシジョンテーブル」タブの上部の条件指定の欄に入力し、下半分のアクションを示す部分には入力せずに「計算ルールテーブル」タブにも内容を記入します。



このルールフロー図を、作成する操作(メソッド)の数だけ定義します。その後、ソースコード生成を実行すると、定義した内容からソースコードを自動生成します。

9. ソースコードの生成

ソースコードの生成方法は、通常のクラス図からのソースコード出力と同じです。以下に、今回の例で出力される Java のソースコードの全てを示します。

(段下げの調整や空行の削除を行っています。)

なお、このソースコード生成の結果(モデルからソースコードを生成するためのルール)はカスタマイズすることが可能になっています。このカスタマイズは「コード生成テンプレート」と呼ばれる Enterprise Architect の拡張の仕組みを利用します。この「コード生成テンプレート」のカスタマイズについてはさまざまな知識が必要となります。

```
public void setRent(Customer customer,Rent rent,Car car)
{
  // behavior is a Activity

  //有効な免許証を持っていないお客様はレンタカーを借りることはできない
  if (customer.validLicenceNumber == "FALSE")
  {
    customer.eligibile = false;
  }
  //18歳未満のお客様はレンタカーを借りることはできない
  else if (customer.age < 18)
  {
    customer.eligibile = false;
  }
  //事故レベルが3のお客様はレンタカーを借りることはできない
  else if (customer.badHistoryLevel == 3)
  {
    customer.eligibile = false;
  }

  if (customer.eligible == true)
  {

    //小型車は1日80ドル
    if (car.type == "Small")
    {
      rent.rentPerDay = 80;
    }
    //4輪駆動車は1日100ドル
    else if (car.type == "AWD")
    {
      rent.rentPerDay = 100;
    }
```

```
    //高級車は1日150ドル
    else if (car.type == "Luxury")
    {
      rent.rentPerDay = 150;
    }

      //レンタカー代は1日あたりの単価と借りる日数で計算される
    rent.rentPayable = rent.rentPerDay * rent.rentDays;

    if (customer.badHistory == true)
    {
      //ペナルティは事故レベルが0の人には適用されない
      if (customer.badHistoryLevel == 0)
      {
        rent.penaltyFee = 0;
      }
      //事故レベルが2のお客様はレンタカー代が20%増し
      else if (customer.badHistoryLevel == 2)
      {
        rent.penaltyFee = rent.rentPayable * 0.2;
      }
      //事故レベルが1のお客様はレンタカー代が10%増し
      else if (customer.badHistoryLevel == 1)
      {
        rent.penaltyFee = rent.rentPayable * 0.1;
      }
    }
    else
    {
    }
      //最終的な総支払額は、レンタカー代の合計と(もしあれば)ペナルティで決定される
      rent.totalAmountPayable = rent.rentPayable + rent.penaltyFee;
  }
  else
  {
  }
}
```

以上で、ビジネスルールからのコード生成機能の概要となります。ぜひご活用ください。