DMN (Decision Model and Notation) : De la Modélisation à l'Automatisation des Décisions

Thierry Biard, Jean-Pierre Bourey, Michel Bigand

DMN (Decision Model and Notation) :

De la Modélisation à l'Automatisation des Décisions

Thierry BIARD[1], Jean-Pierre BOUREY[2], Michel BIGAND[2]

1. Laboratoire Génie Industriel - CentraleSupélec – Université Paris-Saclay
Grande Voie des Vignes, 92290 Châtenay-Malabry, France
thierry.biard@centralesupelec.fr

2. École Centrale de Lille - Université Lille Nord de France
Cité Scientifique - CS 20048, 59651 Villeneuve d'Ascq Cedex, France
jean-pierre.bourey@centralelille.fr, michel.bigand@centralelille.fr

RESUME. Cet article s'intéresse à la nouvelle notation DMN (Decision Model and Notation) qui est utilisée pour modéliser de façon standard les prises de décisions. Après une présentation du contexte, les principaux éléments de DMN sont montrés, puis utilisés dans une étude de cas. Les trois modèles CIM, PIM et PSM de la MDA (Model Driven Architecture) sont rappelés, avant d'être appliqués à DMN. Cette approche permet de comprendre, puis de mettre en œuvre deux solutions techniques différentes pour automatiser les prises de décision, but de cet article. Un chapitre est consacré au changement de paradigme : la programmation déclarative (versus la programmation procédurale). Les propos sont illustrés par du code informatique extrait de notre démonstrateur.

ABSTRACT. This article focuses on the new DMN notation (Decision Model and Notation) which is used to model with a standard way the decision-making. After a context presentation, the main DMN elements are shown, then used into a case study. The three models CIM, PIM and PSM of the MDA (Model Driven Architecture) are reminded, before being applied to DMN. This approach allows to understand then to enforce a couple of different technical solutions to automate decision-making, goal of this article. A chapter is dedicated to the paradigm change: the declarative programming (versus the procedural programming). The talking is illustrated by some computer code extracted from our demonstrator.

MOTS-CLES : DMN, décision, modélisation, notation, règle métier, table de décision, automatisation, processus, BRMS, Drools.

KEYWORDS: DMN, decision, model, notation, business rules, decision table, automation, process, BRMS, Drools.

1. Introduction

Dans un contexte général de modélisation d'entreprise, il apparait que les notations standards utilisées pour la représentation des processus métier sont peu adaptées pour la représentation des prises de décisions opérationnelles. Quelques notations intéressantes, à l'initiative d'éditeurs spécialisés, sont apparues ces dernières années, mais elles étaient propriétaires.

Depuis trois ans, une nouvelle notation nommée DMN (Decision Model and Notation) est venue compléter le bouquet de standards proposés par l'OMG (Object Management Group) dans le cadre de la modélisation métier.

Cette notation DMN permet de modéliser les prises de décision et les règles métier. Son objectif principal est de fournir une représentation de la prise de décision compréhensible par toutes les parties prenantes (acteurs métier qui gèrent et pilotent les décisions, analystes métiers qui spécifient les exigences relatives aux prises de décisions, développeurs responsables de l'automatisation de tout ou partie de la prise de décision).

L'aspect visuel de cette notation DMN, qui ne comporte que quatre éléments graphiques principaux, semble trop simple de prime abord pour permettre l'automatisation des prises de décisions qui ont été modélisées. Le but de cet article est de démontrer qu'au contraire, il est possible d'automatiser les prises de décisions qui ont été modélisées avec la notation DMN. C'est sa contribution principale.

Après avoir positionné la notation DMN par rapport aux autres spécifications standards publiées par l'OMG, les principaux éléments de DMN (diagramme et table de décision) seront présentés. La complémentarité entre la notation DMN et la notation BPMN sera évoquée.

La MDA (Model Driven Architecture), qui applique le principe de Séparation des Préoccupations grâce à ses trois modèles CIM, PIM et PSM, va nous fournir la grille de lecture adéquate pour comprendre, puis mettre en œuvre deux solutions techniques différentes, dites spécifique et générique, pour automatiser les prises de décisions modélisées selon la notation DMN.

Ces deux solutions, réalisées à l'aide des mêmes outils gratuits (pour le monde académique au moins) seront succinctement comparées à partir de la même étude de cas, qui consiste à décider (ou non) de la recevabilité d'un vacataire lors d'un processus de recrutement dans un établissement public d'enseignement supérieur.

Sans surprise, ces deux solutions donneront, à partir du même modèle, le même résultat - une prise de décision identique, mais avec deux approches assez différentes. Le critère principal de comparaison sera celui de l'indépendance aux outils utilisés.

Enfin, la conclusion indiquera si le but de cet article est atteint, puis recommandera une solution technique, avant de présager quelques perspectives.

2. DMN (Decision Model and Notation)

2.1. Présentation du contexte

La notation DMN (Decision Model and Notation) [OMG, 2016] est un nouveau standard de l'OMG (Object Management Group). L'OMG publie des dizaines de spécifications standards, dont la plus connue est UML (Unified Modeling Language) [OMG, 2015c]. La Figure 1 positionne les standards qui nous intéressent ici.

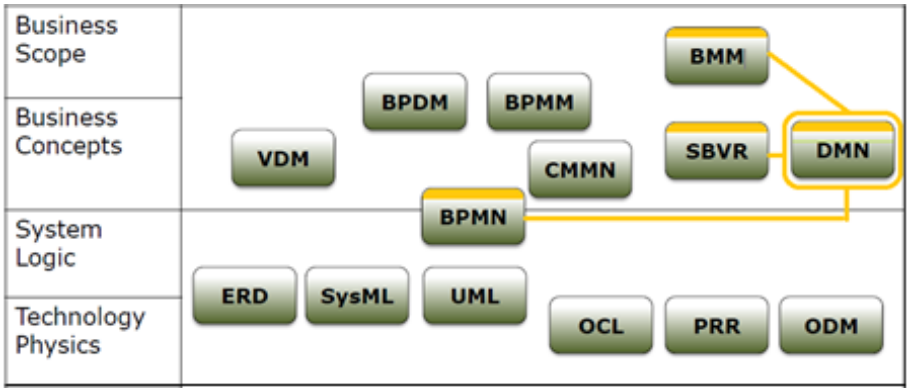

Figure 1: Bouquet de standards de l'OMG, classés selon [Pitschke, 2014]

La notation DMN fait partie des spécifications Business Modeling, avec :

– BMM (Business Motivation Model) [OMG, 2015a], peu connue, car peu utilisée directement, mais qui sert de référence aux autres standards ; une décision représentée en DMN peut supporter un objectif représenté en BMM ;

– SBVR (Semantics of Business Vocabulary and Business Rules) [OMG, 2015b], dont DMN peut utiliser les termes métier qui auront été précédemment définis selon son formalisme [Linehan et de Sainte Marie, 2011] ;

– BPMN (Business Process Model and Notation) [OMG, 2013] ; la notation DMN peut s'utiliser seule ou en complément d'un processus métier représenté en BPMN ; leur complémentarité sera démontrée plus loin.

La première version 1.0 bêta de la spécification DMN a été publiée en février 2014, mais présentait plusieurs lacunes dans son métamodèle (des espaces incongrus dans les noms de classes, voire pas de nom du tout, par exemple), rendant son utilisation difficile. Nous avons néanmoins réussi à développer sur cette base notre propre outil de modélisation DMN [Biard et al., 2015].

C'est la version 1.1, publiée en juin 2016, qui a permis enfin son utilisation réelle et qui sert de base à la plupart des outils disponibles sur le marché aujourd'hui. DMN est donc une notation standard assez récente et encore méconnue.

2.2 Principaux éléments de DMN

2.2.1. DRD (Decision Requirements Diagram)

La représentation emblématique de la notation DMN est le *Decision Requirements Diagram* (DRD). Ce diagramme contient quatre éléments graphiques principaux. Les deux premiers éléments sont obligatoires ; les deux derniers facultatifs :

– *Input Data* (Donnée d'entrée ; au moins une, généralement plusieurs) ;

– *Decision* (Décision ; au moins une ; la Décision est en fait la donnée de sortie).

– *Knowledge Source* (Source de connaissance ; un expert ou un document de référence qui fait autorité dans le métier concerné) ;

– *Business Knowledge Model* (Modèle de connaissance métier, contenant une logique de décision, qui peut être réutilisé dans plusieurs diagrammes DRD).

Trois types de liens différents, les *Requirements*, permettent de représenter les relations entre ces quatre éléments graphiques principaux :

– *Information Requirement*, trait plein terminé par une pointe, qui relie :

- soit une Donnée d'entrée vers une Décision ;

- soit une Décision secondaire vers une Décision principale ;

– *Authority Requirement*, trait pointillé terminé par un point, qui symbolise l'invocation d'une Source de connaissance ;

– *Knowledge Requirement*, trait pointillé terminé par une pointe, qui symbolise l'invocation d'un Modèle de connaissance métier.

Tous ces composants graphiques de DMN (les quatre éléments principaux et les trois liens différents) sont utilisés dans le diagramme DRD de la Figure 2 :

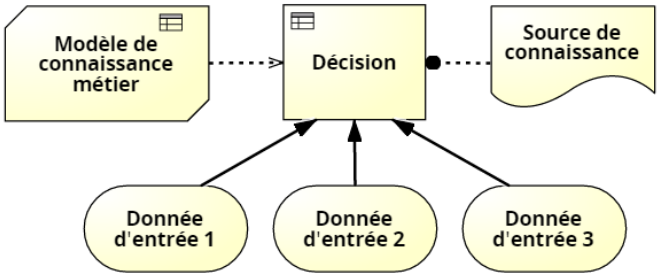

Figure 2: Tous les composants graphiques de DMN dans un diagramme DRD

Les types de liens adéquats pour relier les quatre éléments principaux sont automatiquement sélectionnés par les outils de modélisation, conformément au métamodèle de DMN, fourni par l'OMG. Ces liens sont explicites dans ce métamodèle (extrait en Figure 3), c'est-à-dire représentés par des classes, ce qui n'est pas très courant (les liens-relations sont généralement implicites).

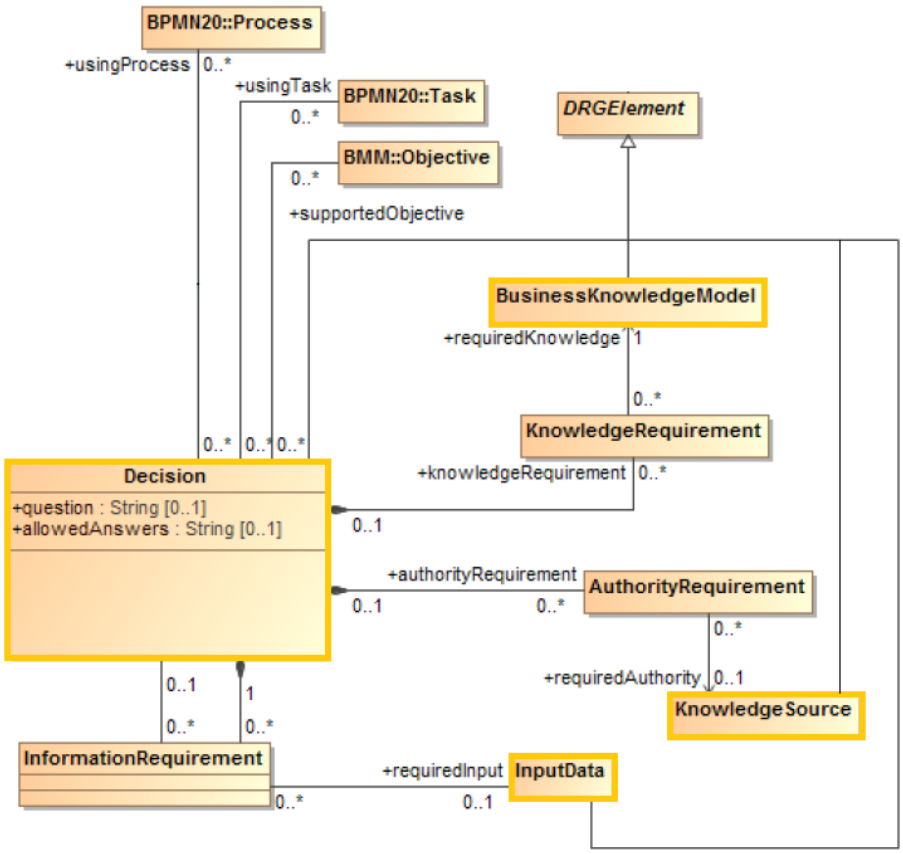

Figure 3: Extrait du métamodèle de DMN de l'OMG (diagramme de classes UML)

Sous une apparente simplicité, le Decision Requirements Diagram peut devenir rapidement complexe (si la prise de décision est elle-même complexe).

Dans l'étude de cas qui va illustrer cet article, il s'agit, dans un établissement public d'enseignement supérieur, de décider de la recevabilité (ou non) d'un vacataire lors du processus de recrutement, selon 11 critères différents.

Le DRD représenté par la Figure 4 reste encore simple. Le choix a été fait de représenter chaque critère comme une donnée d'entrée. Cela permet de voir au premier coup d'œil que les critères sont complètement différents, selon qu'il s'agit d'un poste de chargé d'enseignement ou d'un poste d'agent temporaire.

Un autre choix de modélisation aurait pu être de représenter uniquement le vacataire comme seule donnée d'entrée, les 11 critères étant alors considérés comme ses attributs.

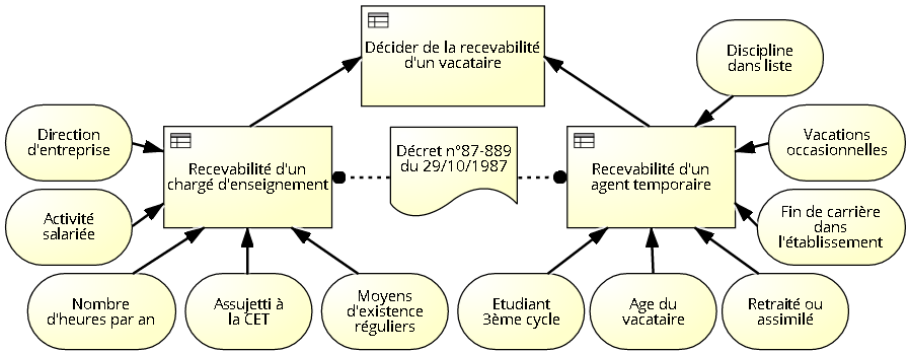

Figure 4: Diagramme (DRD) "Décider de la recevabilité d'un vacataire"

On voit aussi sur cet exemple que la décision principale « Décider de la recevabilité d'un vacataire », représentée par convention tout en haut du DRD, est constituée de deux décisions secondaires « Recevabilité d'un chargé d'enseignement » OU « Recevabilité d'un agent temporaire ».

Le Décret n°87-889 est considéré comme Source de connaissance pour chacune des décisions secondaires. Les règles métier, qu'il convient d'appliquer pour décider de la recevabilité d'un vacataire, proviennent de ce document de référence.

Par contre, cette étude de cas n'utilise pas de Modèle de connaissance métier, car la logique de décision est ici très spécifique et ne pourrait pas être réutilisée ailleurs. Un Modèle de connaissance métier s'utile comme une fonction, que l'on appelle avec des paramètres spécifiques à chaque contexte. Son objectif premier est la réutilisation.

2.2.2. Tables de décision

La notation graphique - relativement simple - du DRD ne permet pas de représenter la logique de décision elle-même. Celle-ci est représentée généralement (surtout lorsque les critères sont relativement nombreux) par une table de décision [Vanthienen et Dries, 1994].

Le formalisme choisi dans la spécification DMN pour représenter les tables de décision est celui du CODASYL, qui existe depuis quelques décennies [Codasyl, 1982]. Le Tableau 1 représente par exemple la logique pour décider de la recevabilité d'un chargé d'enseignement, en fonction de 5 critères en entrée (4 booléens et 1 nombre).

La logique pour décider de la recevabilité d'un agent temporaire est représentée par une table de décision similaire, avec des critères en entrée différents, bien sûr. Quant à la décision principale de recevabilité d'un vacataire, il s'agit d'une table très simple représentant un OU logique des deux décisions secondaires. Ces deux tables ne sont pas reproduites dans cet article, pour des contraintes de place.

Tableau 1: Table de décision "Recevabilité d'un chargé d'enseignement"

Recevabilité d'un chargé d'enseignement						
U	Entrées					Sortie
	Direction entreprise	Activité salariée	Nombre d'heures par an	Assujetti à la CET	Moyens existence réguliers	Recevabilité d'un chargé enseignement
	Booléen	Booléen	Nombre	Booléen	Booléen	Booléen
1	= vrai	-	-	-	-	vrai
2	= faux	= vrai	≥ 900	-	-	vrai
3	= faux	= vrai	< 900	-	-	faux
4	= faux	= faux	-	= vrai	-	vrai
5	= faux	= faux	-	= faux	= vrai	vrai
6	= faux	= faux	-	= faux	= faux	faux

La lettre « U » (comme Unique) positionnée en haut et à gauche de cette table de décision signifie qu'il ne doit pas de recouvrement entre chacune des 6 règles. Unique est la politique de succès (*hit policy*) qui est recommandée : pour chaque jeu de données d'entrée, une seule règle doit s'appliquer.

La formalisation des règles métier est une spécialité à part entière [Ross, 2013]. Les outils spécialisés dans la notation DMN - une vingtaine sont déjà disponibles [OpenRules, 2016] - permettent de vérifier automatiquement la complétude et la cohérence des tables de décision, ce qui est un gage de qualité.

2.2.3. Langage FEEL

FEEL (*Friendly Enough Expression Language*) est un langage déclaratif proposé dans la spécification DMN qui permet de formaliser la logique de décision sous la forme de code informatique, indépendamment de toute plate-forme [Silver, 2016]. Exemple : `if Vacataire.DirectionEntreprise = true then true`

FEEL n'est toutefois pas suffisant pour représenter tous les éléments de DMN (diagrammes et tables de décision), comme cela sera montré plus loin.

2.2.4. Complémentarité de DMN avec BPMN

La notation DMN peut s'utiliser seule ou bien en complément de la notation BPMN [Debevoise et Taylor, 2014]. Chaque Décision peut être utilisée par un processus ou une tâche, comme indiqué sur l'extrait du métamodèle (Figure 3).Nous avons montré dans un précédent article que la notation BPMN se révélait souvent mal adaptée pour représenter des prises de décisions complexes [Biard et al., 2015].

Dans un diagramme BPMN, il suffit alors de remplacer une multitude de branchements en cascade par une seule tâche de type *Business Rules*, symbolisée par une petite table de décision en haut à gauche, pour faire le lien avec un diagramme DMN. La bonne pratique est de nommer à l'identique la tâche du diagramme BPMN et la décision principale du diagramme DMN.

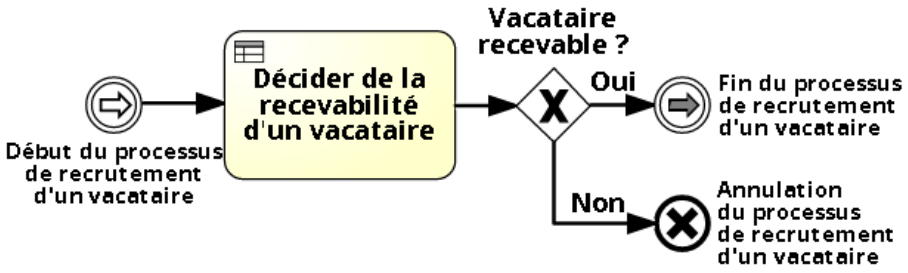

Figure 5: Extrait du diagramme BPMN Processus de recrutement d'un vacataire

Généralement, une tâche de type Business Rules est suivie par un branchement du type OU exclusif, comme dans l'exemple ci-dessus, mais pas toujours. En effet, la notation DMN peut servir à modéliser un calcul complexe, comme un pourcentage de remise en fonction de la fidélité d'un client ou le taux d'un emprunt en fonction des risques présentés par l'emprunteur : dans ce cas, la tâche retourne le résultat du calcul uniquement, qui sera transmis à la tâche suivante.

3. MDA (Model Driven Architecture)

3.1 Trois modèles CIM, PIM & PSM

La *Model Driven Architecture* (MDA) ou l'Architecture Dirigée par les Modèles est un concept proposé par l'OMG également [OMG, 2014]. Ce concept est indépendant de tout langage ou notation. La MDA applique notamment le principe de Séparations des Préoccupations [Dijkstra, 1974].

La MDA propose entre autres trois modèles à différents niveaux, qui semblent particulièrement bien adaptés pour comprendre, voire comparer, les deux solutions proposées pour automatiser les prises de décisions modélisées avec la notation DMN. Le Tableau 2 qui rassemble ces 3 modèles va nous servir de grille de lecture.

Tableau 2: Les trois modèles MDA : CIM, PIM & PSM

Trois modèles MDA	Description
CIM (Computation Independent Model)	**Modèle de représentation du métier, indépendant de toute considération informatique**
PIM (Platform Independent Model)	**Modèle de conception pour l'informatique, indépendant de la plate-forme d'exécution**
PSM (Platform Specific Model)	**Modèle de conception pour l'informatique, spécifique à la plate-forme d'exécution**

3.2. Projection de DMN sur les modèles de la MDA

Si l'on projette les éléments principaux de DMN présentés précédemment sur les trois modèles MDA, on obtient assez logiquement le Tableau 3 dans lequel la plate-forme cible choisie est Drools [Red Hat, 2017], le plus connu des BRMS (Business Rules Management System), qui contient entre autres un moteur de règles. Ce tableau est une première approche théorique. En effet, les expérimentations présentées dans les chapitres suivants démontreront que ce Tableau 3 devra être complété.

Tableau 3: Projection des principaux éléments de DMN sur les modèles MDA

Trois modèles MDA	Principaux éléments de DMN en théorie
CIM (Computation Independent Model)	**DRD (Decision Requirements Diagram) + Tables de décision**
PIM (Platform Independent Model)	**FEEL (Friendly Enough Expression Language)**
PSM (Platform Specific Model)	**DRL (Drools Rule Language)**

4. Démonstrateur pour l'automatisation des prises de décision

4.1. Présentation du démonstrateur

Notre démonstrateur est constitué essentiellement de :

- Signavio Decision Manager version 10.11.0 [Signavio, 2016] pour la modélisation en DMN des prises de décision (diagrammes et tables de décision) ; cet outil fonctionne entièrement en ligne en mode SaaS et ne requiert donc qu'un navigateur web ; il est de bonne facture et d'accès gratuit pour le monde académique. Enfin, il est complètement intégré avec Process Editor pour la modélisation BPMN ;

- Drools (version 7.0.0.Beta6) pour l'automatisation des prises de décisions ; il s'agit du BRMS de référence, qui existe en version gratuite ou payante (sous le nom de Red Hat JBoss BRMS), proposé seul ou intégré dans des logiciels plus complets ; Drools est utilisé ici sous la forme d'un *plug-in* pour Eclipse.

L'environnement de développement intégré Eclipse (version neon.2) a donc été utilisé pour faire fonctionner Drools. D'autres applications comme Git pour la gestion des versions et Maven pour la gestion des dépendances s'avèrent également utiles.

4.2. Première solution spécifique : du CIM au PSM (sans PIM)

4.2.1. Principe de la première solution spécifique

Depuis plusieurs mois, Signavio Decision Manager permet de générer directement un fichier au format DRL (Drools Rule Language), c'est-à-dire du code informatique spécifique à une plate-forme (PSM), à partir d'un diagramme DMN et des tables de décision (CIM) qui lui sont associés.

Le fichier DRL utilise, en plus des bibliothèques standards de Java, une bibliothèque de fonctions spécifiques à Signavio, fournie sous la forme d'un fichier « DMN Formulae Java8-1.0-SNAPSHOT.jar ».

Tableau 4 : Projection des éléments de la première solution sur les modèles MDA

Trois modèles MDA	Signavio Decision Manager (solution 1)
CIM (Computation Independent Model)	**DRD (Decision Requirements Diagram) + Tables de décision**
PIM (Platform Independent Model)	✘
PSM (Platform Specific Model)	**DRL (Drools Rule Language) + DMN Formulae Java8-1.0-SNAPSHOT.jar**

4.2.2 Changement de paradigme : la programmation déclarative

Même si cette solution n'utilise pas le modèle PIM de niveau intermédiaire, il est intéressant de s'y attarder, car le langage DRL ne propose pas moins qu'un changement de paradigme, la programmation déclarative [Van Roy et Haridi, 2004], par rapport à un langage informatique traditionnel (programmation impérative).

En effet, dans le langage DRL, pas de « if…then…else » imbriqués qui rendent le développement, le test et la maintenance compliqués avec les autres langages, mais uniquement un « when…else » pour chaque règle métier, que l'on peut facilement ajouter, modifier ou supprimer en toute indépendance des autres règles.

Un autre avantage majeur est que l'ordre des règles métier n'est pas important (si la politique de succès est de type Unique). Même la boucle « for each » pour prendre en compte tous les objets concernés par les règles métier n'est plus utile en programmation déclarative.

Algorithme 1: Programmation impérative avec "if...then…else"

```
for each objet in mesDonnees do
   if une première condition est détectée
      then une première action est faite
      else
         if une deuxième condition est détectée
            then une deuxième action est faite
            else une troisième action est faite
         endif
   endif
endfor
```

Algorithme 2: Programmation déclarative avec "when...then"

```
rule "une"
   when une première condition est détectée
   then une première action est faite
end
```

```
rule "deux"
   when une deuxième condition est détectée
   then une deuxième action est faite
end
```

```
rule "trois"
   when une troisième condition est détectée
   then une troisième action est faite
end
```

Les vraies règles métier de notre étude de cas s'avèrent assez verbeuses et peu lisibles, comme c'est souvent le cas du code informatique généré automatiquement. L'exemple ci-après (Algorithme 3) a été simplifié (suppression de préfixes), afin que les lignes aient une longueur raisonnable dans cet article.

Dans la section *when*, les Données d'Entrée sont évaluées en lignes 3, 4 et 5. Dans la section *then*, si les critères de décision sont respectés, la Décision RecevabiliteChargeEnseignement devient vraie en ligne 8. A noter que cette Décision est insérée dans la base de faits en ligne 9, car il s'agit d'une sous-décision pour la Décision principale DeciderRecevabilitéVacataire et sa valeur sera utilisée ultérieurement dans une autre règle.

Algorithme 3: Exemple de code DRL simplifié pour la règle n°2 du Tableau 1

```
1: rule "recevabiliteChargeEnseignement_rule_2"
2:   when
3:     eval(nullSafeEval(equals(getDirectionEntreprise(),false)))
4:     eval(nullSafeEval(equals(getActiviteSalariee(),true)))
5:     eval(nullSafeEval(greaterThanOrEqualTo(getNombreHeuresParAn(),
                                      BigDecimal.valueOf(900.0))))
6:   then
7:     RecevabiliteChargeEnseignement $recevabiliteChargeEnseignement
                            = new RecevabiliteChargeEnseignement();
8:     $recevabiliteChargeEnseignement.
                           setRecevabiliteChargeEnseignement(true);
9:     insert($recevabiliteChargeEnseignement);
10:end
```

4.2.3 Déclaration simplifiée des Données d'entrée, sans caractères accentués

Le challenge était de pouvoir utiliser les fichiers au format DRL, tel quel, sans aucune modification, afin qu'un changement de règle puisse être répercuté facilement. Seuls les caractères accentués de la langue française posent problème, car ils sont interdits dans la plupart des langages de programmation et en l'occurrence supprimés lors de la génération des fichiers au format DRL, rendant la compréhension des noms des éléments compliquée. Un diagramme et ses tables de décision modifiés, sans aucun caractère accentué, ont donc été nécessaires. Le langage DRL permet ensuite une déclaration des Données d'entrée très simple, grâce à la directive *declare*.

Algorithme 4: Déclaration des Données d'entrée simplifiée avec le langage DRL

```
1: declare Input
2:    directionEntreprise : Boolean
3:    activiteSalariee: Boolean
4:    nombreHeuresParAn: BigDecimal
5:    assujettiCet : Boolean
6:    moyensExistenceReguliers: Boolean
7: end
```

4.2.4 Application des règles métier pour un vacataire

Afin d'appliquer toutes les règles métier générées automatiquement (ligne 09), il convient d'écrire un programme en langage Java, qui va d'abord créer (ligne 02) puis initialiser (lignes 03 à 07) un objet de type vacataire, puis l'insérer (ligne 08) parmi les faits de la session Drools. Cette partie du programme est relativement simple et lisible. Ce n'est pas le cas du code qui suit (lignes 10 à 13), nécessaire à la récupération du résultat (la décision *output*). Ces quatre lignes de codes ont nécessité beaucoup d'essais, notamment à cause du typage spécifique des objets.

Algorithme 5: Programme Java d'application des règles, première solution (extrait)

```
01: FactType vacataireType =
       kieBase.getFactType("com.signavio.droolsexport.mon_modele_dmn",
          "Input");
02: Object thierry = vacataireType.newInstance();
03: vacataireType.set(thierry, "directionEntreprise", true);
04: vacataireType.set(thierry, "activiteSalariee", false);
05: vacataireType.set(thierry, "nombreHeuresParAn",
       new BigDecimal("800"));
06: vacataireType.set(thierry, "assujettiCet", true);
07: vacataireType.set(thierry, "moyensExistenceReguliers", true);
08: kieSession.insert(thierry);
09: kieSession.fireAllRules();

10: FactType outputType =
       kieBase.getFactType("com.signavio.droolsexport.mon_modele_dmn",
          "DeciderRecevabiliteVacataire_Output");
11: Collection<?> outputObjects = kieSession.getObjects(new
       ClassObjectFilter(outputType.getFactClass()));
12: Object outputObject = outputObjects.iterator().next();
13: boolean output = (boolean) outputType.get(outputObject,
       "deciderRecevabiliteVacataire");
14: System.out.println("Recevabilité du vacataire = " + output);
```

Quant à l'affichage du résultat (la décision *output*), celui-ci s'avère peu spectaculaire dans notre étude de cas, puisqu'il s'agit d'une simple variable booléenne qui représente la recevabilité du vacataire lors de son recrutement : *true* ou *false* :

```
Recevabilité du vacataire : true
```

4.3. Deuxième solution générique : du CIM au PIM (sans PSM)

4.3.1 Principe de la deuxième solution générique

Depuis quelques semaines, Signavio Decision Manager permet également de générer un fichier XML au format DMN version 1.1 (appelé « DMN 1.1 XML »), toujours à partir d'un diagramme DMN et des tables de décision (CIM) qui lui sont associés. « DMN 1.1 XML » est un format d'interchange défini dans la spécification DMN. Un schéma XSD est fourni par l'OMG, permettant de valider le fichier XML généré. Il s'agit donc d'un PIM, car il est indépendant de toute plate-forme.

Le but originel de ce format d'interchange est de pouvoir échanger des modèles de prise de décision entre des outils différents. Ce fichier XML dit sérialisé définit tous les éléments (y compris des métadonnées) nécessaires à la représentation du diagramme et des tables de décision. Ce format d'interchange est également capable de contenir du langage FEEL (qui lui n'est pas suffisant pour tout définir).

La prochaine version 7 de Drools étant capable d'interpréter directement le format « DMN 1.1 XML » (comme quelques autres outils spécialisés dans la modélisation et l'exécution des décisions), la génération de code spécifique à une plate-forme (PSM) devient alors superflue.

Tableau 5: Projection des éléments de la deuxième solution sur les modèles MDA

Trois modèles MDA	Signavio Decision Manager (solution 2)
CIM (Computation Independent Model)	**DRD (Decision Requirements Diagram) + Tables de décision**
PIM (Platform Independent Model)	**DMN 1.1 XML (Modèle interchangeable) contenant éventuellement du langage FEEL**
PSM (Platform Specific Model)	✘

4.3.2 Application des règles métier pour un vacataire

Le fichier « DMN 1.1 XML » généré par Signavio Decision Manager n'a pas fonctionné tel quel avec Drools. Quelques ajustements ont été nécessaires, essentiellement l'ajout du namespace « signavio » comme préfixe des types de variables créées avec cet outil : `<variable typeRef="signavio:monType"` (tandis que les types de variables génériques sont déjà définis dans le namespace « feel » : `<typeRef>feel:boolean</typeRef>`). Le préfixe adéquat sera sans doute ajouté dans d'une prochaine version de l'outil.

Afin d'appliquer les règles métier générées automatiquement, il convient également d'écrire un petit programme en langage Java. Alors qu'il faut ajouter au préalable deux lignes de code pour prendre le fichier « DMN 1.1 XML », la création et l'initialisation du contexte sont très similaires à celui d'un objet. Par contre, la méthode de récupération du résultat (toujours aussi peu spectaculaire) s'avère particulièrement concise, voire élégante, par rapport à celle de la première solution.

Algorithme 6: Programme Java d'application des règles, deuxième solution (extrait)

```
01: DMNRuntime runtime = DMNRuntimeUtil.createRuntime(
       "13-Decider-recevabilite-vacataire-DMN.dmn", this.getClass() );
02: DMNModel dmnModel = runtime.getModel(
       "http://www.signavio.com/dmn/1.1/diagram",
          "13-Decider-recevabilite-vacataire-DMN" );

03: DMNContext context = DMNFactory.newContext();
04: context.set( "directionEntreprise", true);
05: context.set( "activiteSalariee", false);
06: context.set( "nombreHeuresParAn", 800);
07: context.set( "assujettiCet", true);
08: context.set( "moyensExistenceReguliers", true);

09: DMNResult dmnResult = runtime.evaluateAll( dmnModel, context );
10: DMNContext result = dmnResult.getContext();
11: System.out.println( "Recevabilité du vacataire : " + result.get(
       "deciderRecevabiliteVacataire" ) );
```

4.3. Comparaison des deux solutions

La deuxième solution générique a sans aucune hésitation notre préférence. Elle est mieux alignée sur les modèles MDA que la première solution spécifique car cette deuxième solution est indépendante de toute plate-forme. Elle est très intéressante si la plate-forme cible - le moteur de règles (BRMS) - est capable d'interpréter directement des fichiers au format « DMN 1.1 XML » (ce qui reste encore assez rare).

En fait, la première solution a permis de mettre en valeur la deuxième ! La première solution a également permis de découvrir le changement important de paradigme, la programmation déclarative, qui est également utilisée de manière implicite dans la seconde solution.

5. Conclusion et perspectives

Malgré son aspect apparemment simpliste, nous avons démontré dans cet article que les prises de décision modélisées en notation DMN (diagramme et tables de décision) peuvent être automatisées. Les différents modèles de la MDA nous ont permis de porter un regard d'architecte sur les solutions techniques proposées, afin de les comprendre puis de les mettre en œuvre.

Cette mise en œuvre nécessite aussi des compétences de développeur informatique. Toutefois, avec seulement quelques lignes de code Java (de niveau avancé), il est possible d'appliquer des règles métier complexes, qui ont été définies en amont par des analystes métier. Si le jeu de données d'entrée (les critères de décision) est complet dès le départ, il est possible de modifier les règles sans changer le code qui les applique.

Tandis que l'automatisation des règles métier existe depuis une quinzaine d'années [Taylor, 2007], le fait de pouvoir désormais s'appuyer en amont sur la notation standard DMN est un progrès indéniable.

Après que les processus métier aient été extraits des applications au début des années 2000, nous pouvons présager que la prochaine étape sera l'extraction des règles métier dans les années qui viennent. L'existence de la notation standard DMN pour la modélisation des prises de décisions, et la possibilité récente d'automatiser ces prises de décisions constituent de sérieux atouts pour la réalisation de ce présage.

6. Bibliographie et références

Biard, T., Le Mauff, A., Bigand, M., Bourey, J.-P. (2015). Separation of Decision Modeling from Business Process Modeling Using New « Decision Model and Notation » (DMN) for Automating Operational Decision-Making. In L. M. Camarinha-Matos, F. Bénaben, & W. Picard (Éd.), *Risks and Resilience of Collaborative Networks* (Vol. 463, p. 489-496). Springer International Publishing.

Codasyl. (1982). A modern appraisal of decision tables. ACM.

Debevoise, T., Taylor, J. (2014). *The MicroGuide to Process and Decision Modeling in BPMN/DMN*. ACR.

Dijkstra, E. W. (1974). On the role of scientific thought.

Linehan, M., de Sainte Marie, C. (2011). The Relationship of Decision Model and Notation (DMN) to SBVR and BPMN. http://www.brcommunity.com/b597.php

OMG. (2013). Business Process Model and Notation (BPMN). http://www.omg.org/spec/BPMN/

OMG. (2014). MDA (Model Driven Architecture) Specifications. http://www.omg.org/mda/specs.htm

OMG. (2015a). Business Motivation Model (BMM). http://www.omg.org/spec/BMM/

OMG. (2015b). Semantics of Business Vocabulary and Rules (SBVR). http://www.omg.org/spec/SBVR/

OMG. (2015c). Unified Modeling Language (UML). http://www.omg.org/spec/UML/

OMG. (2016). Decision Model and Notation (DMN). http://www.omg.org/spec/DMN/

OpenRules. (2016). Decision Model and Notation (DMN) Supporting Tools. http://openjvm.jvmhost.net/DMNtools/

Pitschke, J. (2014). Mastering Business Modeling – Applying Business Architecture and Standards successfully. http://www.enterprise-design.eu/files/images/download-praesentaionen/BPMEurope2014_JPitschke.pdf

Red Hat. (2017). Drools, Business Rules Management System. http://www.drools.org

Ross, R. G. (2013). *Business rule concepts: getting to the point of knowledge* (4th Ed). Business Rule Solutions.

Signavio. (2016). Decision Manager. http://www.signavio.com/products/decision-manager/

Silver, B. (2016). *DMN method and style*. Cody-Cassidy Press.

Taylor, J. (2007). *Smart (enough) systems: how to deliver competitive advantage by automating the decisions hidden in your business.* Prentice Hall.

Van Roy, P., Haridi, S. (2004). *Concepts, techniques, and models of computer programming*. MIT Press.

Vanthienen, J., Dries, E. (1994). Decision Tables: Refining the Concept and a Proposed Standard. *ResearchGate*.