Systèmes experts et IA :

des perspectives toujours plus vastes pour les entreprises avec le moteur d'inférence Drools

THIERRY GUERIN

SENIOR MIDDLEWARE ARCHITECT

onepoint.
beyond the obvious

Systèmes experts et IA : des perspectives toujours plus vastes pour les entreprises avec le moteur d'inférence Drools

Introduction

Depuis quelques années, l'intelligence artificielle est devenue pour beaucoup synonyme de machine learning. Le battage médiatique autour de la révolution de l'intelligence artificielle y est sans doute pour quelque chose.

Pourtant, l'intelligence artificielle est un domaine bien plus vaste. En effet, le machine learning n'est qu'une des approches de cette matière, approche que l'on appelle également « approche statistique ».

Dans le domaine de l'IA, il existe deux grandes familles : d'un côté l'approche statistique (parfois aussi appelée probabiliste), et de l'autre l'approche déterministe. Aucune de ces deux approches n'est supérieure à l'autre, elles font chacune appel à des procédés différents et sont simplement plus ou moins adaptées selon les différents cas d'usage. Fondamentalement, les systèmes d'intelligence artificielle ont en commun d'être conçus pour imiter des comportements propres aux humains et permettent la résolution de problèmes complexes à résoudre.

Depuis quelques années, l'intelligence artificielle est devenue pour beaucoup synonyme de machine learning. Le battage médiatique autour de la révolution de l'intelligence artificielle y est sans doute pour quelque chose.

Pourtant, l'intelligence artificielle est un domaine bien plus vaste. En effet, le machine learning n'est qu'une des approches de cette matière, approche que l'on appelle également « approche statistique ».

Dans le domaine de l'IA, il existe deux grandes familles : d'un côté l'approche statistique (parfois aussi appelée probabiliste), et de l'autre l'approche déterministe. Aucune de ces deux approches n'est supérieure à l'autre, elles font chacune appel à des procédés différents et sont simplement plus ou moins adaptées selon les différents cas d'usage. Fondamentalement, les systèmes d'intelligence artificielle ont en commun d'être conçus pour imiter des comportements propres aux humains et permettent la résolution de problèmes complexes à résoudre.

1. Comprendre les systèmes experts

Les systèmes experts font partie d'une branche spécifique de l'intelligence artificielle appelée « déterministe ». Leur objectif commun est d'imiter le raisonnement d'un professionnel spécialiste dans un domaine précis comme en matière de crédits immobiliers ou de diagnostic médical. Les systèmes experts s'appuient sur la connaissance d'un domaine spécifique, préalablement et consciencieusement, recueillie auprès d'un spécialiste d'un domaine métier (expert humain).

Un système expert est composé de trois grands modules :

- Une ou plusieurs base(s) de règles spécifiques au domaine métier : les connaissances sous une forme déclarative ;
- Des données d'entrées nommées « base de faits » qui servent au raisonnement et auxquelles s'appliquent les règles logiques ;
- Un moteur d'inférence qui fait tourner les règles et des données d'entrées qui servent au raisonnement.

L'ensemble constitué d'un modèle de faits et de la base de règles est aussi appelé la base de connaissance.

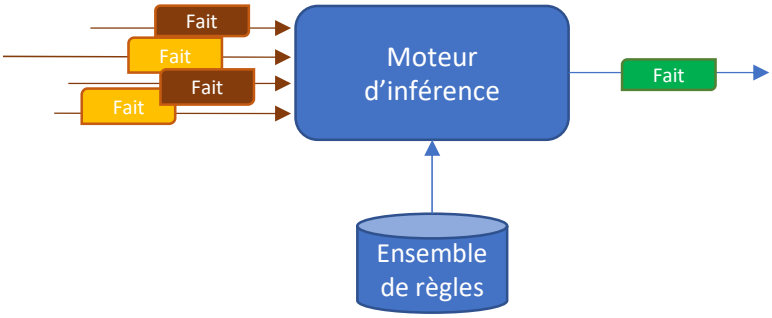


Figure 1 - Architecture d'un Système Expert

Le moteur d'inférence est capable d'utiliser faits et règles pour produire de nouveaux faits, jusqu'à parvenir à un objectif ou à la réponse à la question posée initialement.

La définition des règles pertinentes dépend des algorithmes de résolution inclus dans le moteur d'inférence. Les plus couramment utilisés sont le chaînage avant (partant des faits pour en rechercher les conséquences) et le chaînage arrière (partant des conclusions envisagées pour vérifier si les conditions sont réunies).

Toute la difficulté de la mise en place d'un système expert réside dans l'acquisition des connaissances d'un expert et de leur formalisation en tant que faits (données manipulées) et règles.

Les systèmes experts possèdent un certain nombre de points forts. Tout d'abord, l'encapsulation (ou la représentation) de la connaissance sous la forme de règles logiques permet la maintenance de la base de règles, ensuite, cela favorise la transmission du savoir ou l'extension à d'autres domaines (par l'ajout de règles plus spécifiques si nécessaire).

Ensuite, le système expert peut expliquer son raisonnement (« quelles règles ont été appliquées ? »), ce qui permet à l'utilisateur de juger de son niveau de fiabilité (ou d'invalider un raisonnement). L'enjeu éthique de la transparence des moteurs d'IA est d'ailleurs surveillé par la CNIL

La mise au point d'un système expert peut être réalisé rapidement et est d'une grande fiabilité. Il est maitrisé par ses concepteurs.

Enfin les systèmes experts peuvent raisonner sur des données partiellement connues avec de la logique floue par l'utilisation de coefficients ou poids associés à certaines règles, c'est-à-dire des connaissances approximatives ou incertaines comme « la température du four est très élevée », « l'enfant est grand », « le contrat est important » … Mais si le contrat est important à partir de 500K€, est-il moins important à 499K€ ? L'utilisation de coefficients de vraisemblance permet de prendre en compte cet aspect.

Mais les systèmes experts ont aussi des limitations. Ainsi, un expert humain est nécessaire. Il doit pouvoir coopérer au projet et pouvoir expliquer son raisonnement alors que de nombreux experts éprouvent des difficultés à les expliciter car le savoir métier est peut-être perdu ou non documenté.

Le domaine d'expertise du système expert est très étroit et est limité aux frontières du domaine métier.

Enfin, les systèmes experts, contrairement aux technologies plus récentes de l'IA (Machine Learning) ne permettent pas l'apprentissage et ne gèrent pas la reconnaissance d'images ou de la parole, ce qui limite leurs portées.

Système Expert	Machine learning
Basé sur des règles et sur un modèle de faits Nécessite l'identification de toutes les règles : aucune erreur ou omission permise.	Basé sur les données Nécessite la confrontation à des milliers/millions de cas pour renforcer la robustesse des décisions. Requiert la disponibilité d'un large volume de données pertinentes et de bonne qualité.
Algorithmes définis et maitrisés par l'homme Évolution au gré des concepteurs.	Algorithme engendré par la machine. Évolution continue selon l'apprentissage.
Délai de conception dépendant du type d'entrants : spécialement bien adapté aux tables de décision. Développement rapide	Mise en œuvre longue. L'identification des données utiles et leur contrôle nécessitent une équipe d'experts en science des données (data scientists et data engineers). Investissement dans l'apprentissage.
Grande fiabilité. Résultats stricts, rigoureux (hors maladresse dans l'établissement des règles). Fourni une trace des raisonnements effectués	Risque d'erreur. Risque de décisions erronées (le système est à la mesure des cas appris et peut créer des faux positifs). Raisonnement de type boite noire : aucune trace

2. Acteurs dans le domaine des Systèmes Experts

Les outils les plus connus dans ce domaine sont bâtis avec des solveurs qui exploitent des bases de règles formelles et des faits. Ces moteurs de règles s'appellent BRMS (Business Rules Management System) ou DMS (Decision Management System) et permettent l'exécution des Systèmes Experts.

On trouve sur ce marché des solutions propriétaires et Open Source. Parmi ces acteurs, on peut citer l'offre commerciale d'IBM ODM (Operational Decision Manager), ou la solution Open Source la plus connue : Drools.

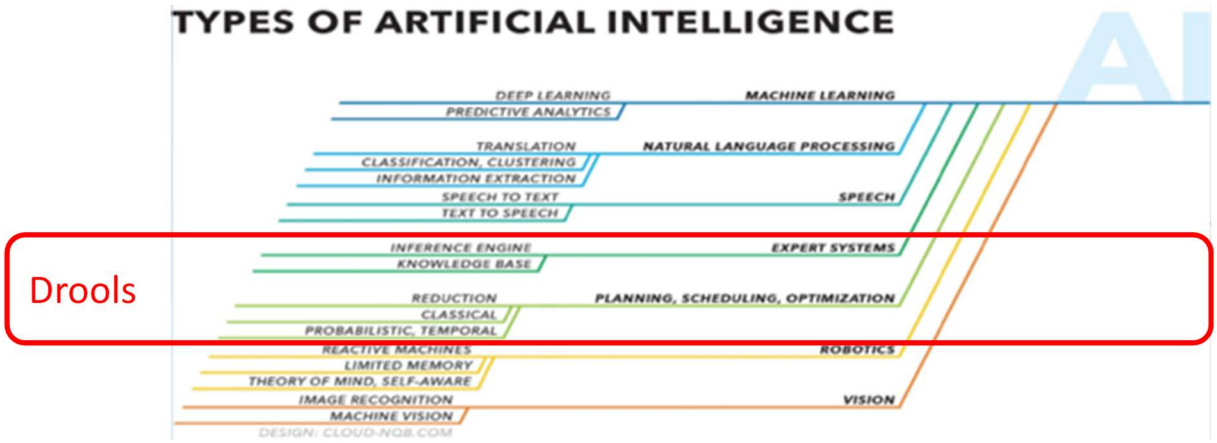


Figure 2 – Positionnement de Drools parmi les différents types d'IA

Drools est une suite logicielle de type « moteur de règles » utilisant un moteur d'inférence à chaînage avant. Drools est 100% Java, Open Source et en licence Apache, compatible avec la plupart des systèmes d'exploitation capables d'exécuter une machine Java virtuelle (JVM), dont Windows, UNIX et Linux.

Red Hat sponsorise le projet communautaire Drools et emploie ses principaux développeurs. Pour les entreprises, Red Hat commercialise sa solution nommée RH-DM (Red Hat Decision Manager), basée sur Drools, avec un système de souscriptions annuelles pour disposer d'un support complet du produit. Les différences entre les deux produits sont liées à ce support et au packaging des produits. Elles sont détaillées sur le site de Drools

Drools fournit la capacité de définir, déployer, exécuter, monitorer et maintenir l'ensemble de règles métiers d'un système, d'une organisation ou d'une entreprise. Le système se compose d'un référentiel de type Git, logiciel de gestion de versions décentralisé, permettant de stocker les règles et d'un ensemble d'outils adaptés aussi bien aux développeurs qu'aux experts fonctionnels pour maintenir la base de connaissance (modèle de faits + règles), ainsi que d'un environnement d'exécution adapté permettant aux applications d'interroger le moteur de règles et d'exécuter les règles.

Voici des exemples de réalisation de systèmes experts : instruction de dossiers de crédits immobiliers, lutte contre la fraude et le blanchissement d'argent (KYC, KYI…), calcul de primes, de retraites, de prix de billets de train, plateformes de jeux, etc...

Drools fournit un ensemble de fonctionnalités : un moteur d'inférence, un environnement d'exécution DMN (Decision Model and Notation), une prise en charge de PMML (Predictive Model Markup Language) qui est une sérialisation XML d'un modèle entrainé de machine learning, ainsi qu'un moteur de traitement d'événements complexes (CEP – Complex Event Processing). En plus de ces fonctionnalités, il intègre aussi une plate-forme d'optimisation de ressources : le solveur de contraintes basé sur le très populaire projet communautaire OptaPlanner.

Drools fournit un environnement d'exécution qui s'adapte facilement à des centaines de milliers de règles dans un seul environnement d'exécution de règles, tout en assurant une exécution à faible latence et hautes performances.

3. Comprendre le principe des « Règles »

Pour identifier les connaissances d'un expert métier, il faut l'interroger de manière à « capturer » son savoir. Il s'agit là d'un exercice délicat car les procédures de résolution lui apparaissent comme « évidentes », alors que l'objectif est de lui faire expliciter les éléments essentiels de sa démarche. Il peut s'agir d'un travail long et difficile selon le domaine et l'expert. C'est le travail du cogniticien de construire une base de connaissances avec les connaissances du domaine et de l'expert.

Toutes les règles doivent être énoncées, sans ordre particulier. Mais la seule condition pour être efficace est de ne pas en oublier une seule. Le système enchainant chaque règle produira de nouvelles assertions qui déclencheront en cascade d'autres règles. Finalement, le système trouvera de lui-même un résultat : ce mécanisme de déduction est appelé moteur d'inférence.

Les systèmes experts reposent sur un principe simple : un raisonnement est une suite d'utilisations de connaissances élémentaires, indépendantes les unes des autres, du type « si je suis dans telle situation alors je fais telle action ». Une règle métier est une déclaration en deux parties : une condition et une action.

Les règles peuvent être ensuite assemblées entre elles au travers de la notation DMN ou d'un processus décrit avec la notation BPMN pour former un raisonnement qui peut être différent pour chaque problème posé.

Drools possède plusieurs méthodes pour écrire des règles métiers dont l'une est d'utiliser son langage propriétaire éponyme : il s'agit d'un langage déclaratif. Voici un exemple qui reprend « Si la condition est remplie, alors l'action se déroule ».

```
when
    <La condition est vraie>
then
    <Réaliser l'action souhaitée>
```

Exemple :

```
rule "promo 1"
when
    cmd : Commande( codePromo == "xyz123" )
then
    modify(cmd) { remise = new BigDecimal("10.00"); }
end
```

Cette règle fixe le montant de la remise à 10 euros si le code promotionnel est égal à "xyz123". La règle comprend deux parties :

1. La condition « when » : s'il y a une commande avec un code promotionnel égal à "xyz123"
2. L'action « then » : modifie la commande en mettant une remise égale à 10 euros.

Qu'est-ce qu'une table de décision ?

Beaucoup de règles peuvent être décrites sous la forme d'un tableau, donc d'une table de décision. Elles sont idéales lorsqu'il existe des règles similaires avec des valeurs différentes.

Une représentation en tableau pour des règles permet de définir chaque colonne comme une condition ou une action, chaque ligne devient alors une règle distincte.

La factorisation des règles est rapide car il suffit de modifier une définition de colonne pour modifier un groupe de règles connexes.

Cette vision est synthétique et compréhensible par tous les acteurs du projet (expert métier, cogniticien, IT…).

Table des catégories de risque				
U	Client existant	Taux du risque de la demande	Score Crédit	**Catégorie du risque**
1	NON	<120	< 590	FORT
2			[590..610]	MOYEN
3			>610	FAIBLE
4		[120..130]	< 600	FORT
5			[600..625]	MOYEN
6			>625	FAIBLE
7		>130	-	TRES FAIBLE
8	OUI	<= 100	< 580	FORT
9			[580..600]	MOYEN
10			>600	FAIBLE
11		>100	< 590	FORT
12			[590..615]	MOYEN
13			>615	FAIBLE

Figure 3 - Exemple de table de décision utilisé en DMN

Il n'est pas rare de rencontrer des tables de décision allant jusqu'à plusieurs centaines de lignes.

Des règles pour le traitement d'événements complexes (CEP – Complex Event Processing)

Les données traitées par le moteur de règles peuvent être perçus comme des événements et pas uniquement comme de simples faits. La capacité à identifier les relations temporelles entre les événements permet alors de mettre en œuvre des scénarios tels que la détection de fraude où une combinaison d'événements apparemment sans intérêt peut être associée à un comportement suspect.

Concevoir son modèle de faits de façon optimale

La représentation des connaissances est un point crucial : la possibilité de représenter les connaissances de l'expert dans le système est le point de passage obligé de toute construction d'un système expert.

Un moteur de règles est, à bien des égards, similaire à une base de données : certaines bonnes pratiques concernant l'utilisation des bases de données s'appliquent également aux systèmes experts. L'une des plus importantes est la conception soigneuse du modèle de données (faits). La qualité de ce modèle est directement proportionnelle à la performance et à la maintenabilité des règles. Un modèle de domaine mal conçu affecte non seulement l'exécution du moteur d'inférence, mais augmente également les temps et les coûts de développement (règles plus complexes à créer et plus difficiles à maintenir dans le temps). Un bon modèle de domaine représente les relations entre les entités métier de la manière la plus simple possible. Les modèles plats apportent généralement davantage en efficacité car ils rendent les contraintes plus faciles à écrire.

Une autre méthode pour décrire un raisonnement avec la norme DMN

DMN, qui signifie Decision Model and Notation, est une norme relativement nouvelle (2015) gérée par OMG, l'organisation derrière BPMN et UML. Elle fait la promotion de DMN pour décrire les règles dans un formalisme indépendant d'une solution logicielle et d'un éditeur.

Drools est le premier environnement d'exécution DMN open-source qui prend en charge le niveau de conformité complet de la spécification DMN.

Structurer un traitement avec la norme BPMN

BPMN (Business Process Model and Notation) est une norme définie par l'OMG pour la spécification des processus métier à l'aide d'un standard de notation lisible pour les développeurs et les concepteurs métier.

Comme l'objectif du moteur de règles n'est pas de gérer des processus métiers qui s'exécuteraient de manière asynchrone mais des processus de décision dont l'exécution complète est immédiate et synchrone, une partie limitée des éléments graphiques de cette norme BPMN est disponible pour définir l'ordonnancement et le type de règles métiers à appliquer :

- Evénements de début et de fin ;
- Activités métier et script ;
- Sous-processus ;
- Passerelles OU exclusives (XOR).

Le choix des connaissances (règles) et leur ordre d'utilisation forment ce que l'on appelle le contrôle du raisonnement.

Cette approche utilise un standard international pour structurer graphiquement les étapes du raisonnement, comblant ainsi le fossé de communication entre la documentation et la mise en œuvre. Les équipes (AMOA, MOE) sont ainsi formées à une norme reconnue, qui est d'ailleurs aussi utilisée par les moteurs de workflow.

Le diagramme suivant représente un processus de décision :

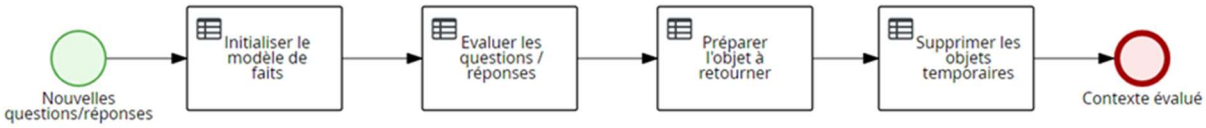
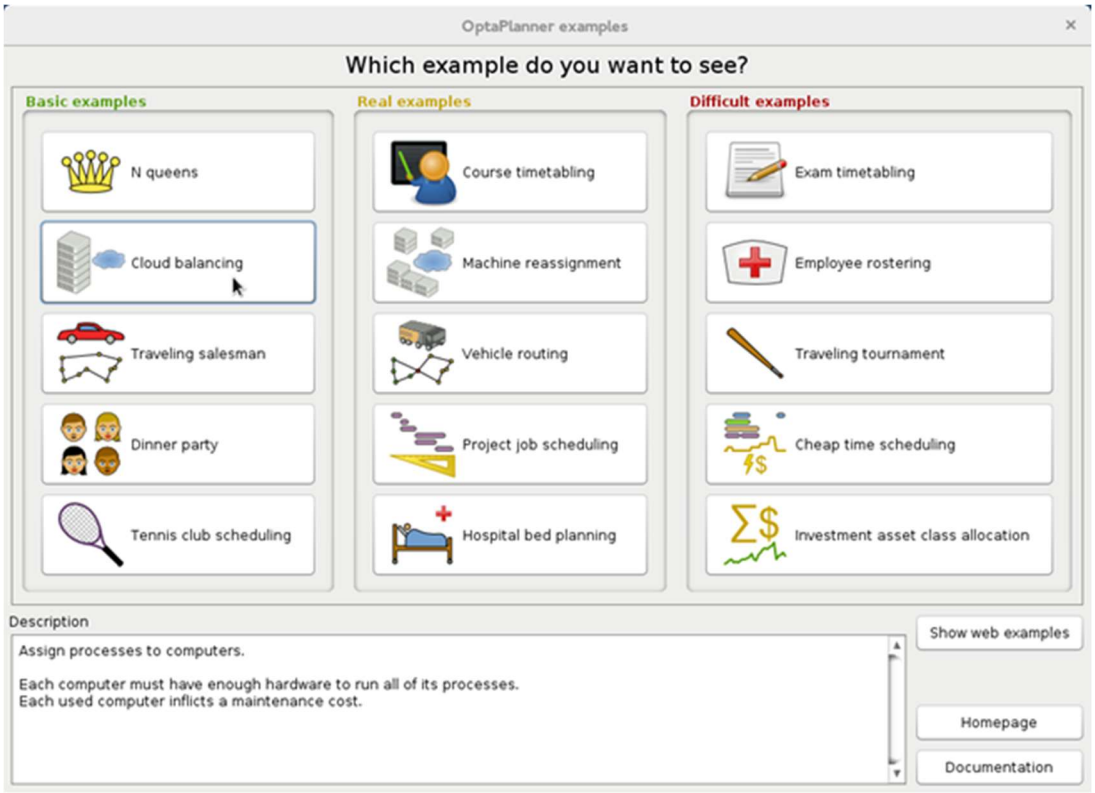


Figure 4 – Exemple de diagramme BPMN pour un Système Expert

Le cercle vert représente le début du traitement. L'exécution se déroule dans le sens des flèches jusqu'à atteindre le cercle rouge, qui représente la fin du raisonnement.

3. L'autre facette de Drools : l'optimiseur de ressources OptaPlanner

Doté de modèles prêts à l'emploi et d'applications de référence, Drools fournit une excellente plate-forme pour automatiser et optimiser toutes sortes de problèmes de planification.



Voici quelques cas d'utilisation : planification des ressources commerciales, optimisation de trajets de véhicules de livraison, gestion du stationnement d'une flotte automobile, planning des employés, planning des cours pour les élèves et professeurs, optimisation de ressources dans le cloud, attribution de tâches, planification de travaux...

Les problèmes de planification sont des problèmes dans lesquels il faut optimiser les objectifs, avec des ressources limitées, sous contraintes. Cette catégorie de problèmes est extrêmement difficile à résoudre et nécessite une IA sophistiquée sous la forme d'algorithmes d'optimisation heuristique pour trouver la solution optimale à ces problèmes dans un temps limité et avec une quantité finie de ressources de calcul.

OptaPlanner fournit un environnement facile à utiliser pour créer des solveurs en Java afin de trouver des solutions optimisées à ces problèmes. Il met la puissance de l'IA entre les mains des développeurs Java qui peuvent utiliser leurs compétences existantes pour créer des applications et des systèmes d'optimisation complexes.

4. Voici un exemple de résolution de problème à l'aide d'un moteur de règles

Il y a quatre golfeurs alignés de gauche à droite.

Le golfeur à droite immédiate de Fred porte un pantalon bleu ;

- Joe est deuxième dans la ligne ;
- Bob porte un pantalon à carreaux ;
- Tom n'est pas en position une ou en position quatre, et il ne porte pas un pantalon orange.

Donnez la liste des joueurs et leurs caractéristiques : nom, position et couleur de pantalon.

Mais avant de regarder la réponse, essayez de résoudre ce problème…

Solution

Notre modèle de données sera composé d'un type de faits nommé Golfer dont les attributs sont :

- `name : String`
- `position : int`
- `color : String`

La règle ci-dessous définira les contraintes dans sa clause « when » :

```
// il existe un golfeur nommé Fred
fred : Golfer ( name == "Fred" )

// Joe est en position 2
joe : Golfer ( name == "Joe",
                position == 2 && != fred.position,
                color != fred.color )

// Bob est habillé d'un patalon à carreau
bob : Golfer ( name == "Bob",
                position != fred.position && != joe.position,
                color == "carreau" && != fred.color && != joe.color )

// Tom n'est pas en position une ou en position quatre,
// et il ne porte pas un pantalon orange
tom : Golfer ( name == "Tom",
                position not in (1, 4, fred.position, joe.position, bob.position),
                color not in ("orange", fred.color, joe.color, bob.color))

// Le golfeur à droite immédiate de Fred porte un pantalon bleu
Golfer ( position == (fred.getPosition() + 1),
          color == "bleu",
          this in ( joe, bob, tom ))
```

La partie « action » de cette règle contiendra l'affichage du résultat sur la console :

```
then
   System.out.println("La position de Fred :" + fred.getPosition()+ ", sa couleur est " + fred.getColor());
   System.out.println("La position de Joe :" + joe.getPosition()  + ", sa couleur est " + joe.getColor());
   System.out.println("La position de Tom :" + tom.getPosition()  + ", sa couleur est " + tom.getColor());
   System.out.println("La position de Bob :" + bob.getPosition()  + ", sa couleur est " + bob.getColor());
end
```

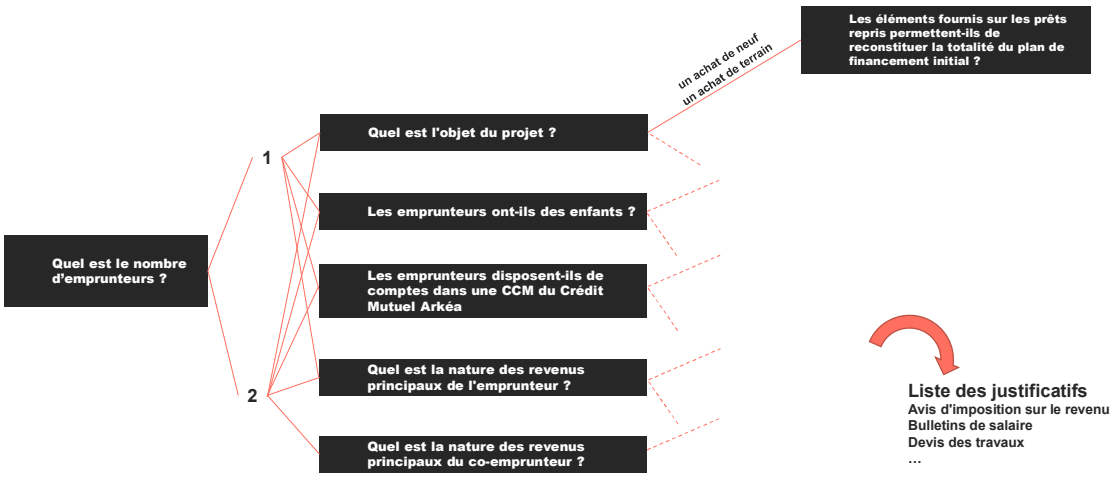
Résultats d'exécution :

- La position de Fred : 1, sa couleur est orange
- La position de Joe : 2, sa couleur est bleue
- La position de Tom : 3, sa couleur est rouge
- La position de Bob : 4, sa couleur est carreau

5. Success Story : un Système Expert en crédits immobiliers

Récemment, pour un de nos clients dans le secteur bancaire, nous avons réalisé un système expert permettant de définir les justificatifs nécessaires pendant le processus d'instruction des dossiers de crédits immobiliers : il comprenait un graphe de 80 questions liées entre elles et 175 justificatifs possibles.

Les données du dossier de crédit immobiliers étaient initialement fournies au Système Expert. Celui-ci avait ensuite la capacité de poser des questions complémentaires à l'utilisateur, ses réponses orientaient alors le système qui, au fur et à mesure, affinait son résultat.



Onepoint a mis à disposition les compétences suivantes :

- Un expert des solutions Drools / RH-DM : pour la définition des best practices, la définition du modèle de faits, l'identification des règles et leur écriture, le paramétrage et le déploiement.
- Un architecte Ops spécialiste des solutions Redhat : pour former les équipes Ops et pour installer la plateforme dans les environnements de développement et de recette.
- Un recetteur chargé des tests : pour contrôler les tables de décision et établir un jeu de tests
- Un pilote de projet

Le déploiement du Système Expert en environnement de recette s'est effectué après un développement rapide de 3 à 4 semaines.

6. Quand utiliser un moteur de règles ?

Il existe une question qui est posée dans presque toutes les discussions à propos de l'utilisation d'un moteur de règles et donc de la mise en œuvre de systèmes experts :

> *Existe-t-il des retours d'expériences pour déterminer l'éligibilité de la mise en œuvre d'un moteur de règles ?*

Eh bien, c'est un sujet, qui peut être traité avec l'utilisation de règles, sous la forme d'une table de décision… Tout comme le ferai un système expert lui-même !

Les règles proposées pour répondre à cette question ne sont pas seulement basées sur des aspects techniques, mais aussi sur l'aspect organisationnel. En fait, l'aspect organisationnel est souvent plus important que les moyens techniques.

Le tableau ci-dessous peut vous aider dans le choix d'utiliser, ou non, un moteur de règles.

Quand	ET	Bénéfices Rapides	Bénéfices à longs termes	Programmation classique	Commentaires
Les spécifications sont disponibles sous la forme de :	Tableaux Excel	+++	+++	- -	La plupart des BRMS (dont Drools) utilisent des Tables de décision.
	If / Then (/ Else) Si / Alors (/Sinon)	++	++	+	"If" peut devenir "when" ("else" est à proscrire)
	Langage Naturel (DSL)	+	+	-	Quelques BRMS (comme Drools) permettent d'exprimer des règles en Langage Naturel
BRMS déjà en place	& disponibilité d'une équipe en support	+++	+++		On y va !
	& une future équipe en support	++	+++		Utilisez ce que vous avez déjà, avant la montée en compétence de l'équipe.
	& aucune équipe prévue en support	++	+		La maintenance à long terme est compromise, surtout si de mauvais choix initiaux ont été adoptés
Pas de BRMS déjà en place	& disponibilité d'une équipe en support	++	+++		L'équipe de soutien va d'abord mettre en place le BRMS.
	& une future équipe en support est prévue	- - - -	++		Vous devrez attendre que le savoir-faire soit disponible ... Ne pas mettre en danger un projet en inscrivant l'implémentation du BRMS sur son chemin critique.
	& aucune équipe prévue en support	- - - -	- - - -		Pas d'infrastructure + un problème de ressource : on n'y va pas !!!
Besoins de	Déploiement à chaud	++	+++	- -	Le déploiement de règles est généralement plus facile qu'une application
	Compréhension / Traçabilité	+++	++	- - -	Permet de suivre le raisonnement
	Modifications rapides de code	++	++	-	Les règles représentent le code métier
	Tests unitaires	++	++	-	Un ensemble de règles peut être testé unitairement.
Evolution d'une application existante n'utilisant pas un BRMS		- - - -	-	+++	
Les règles métier changent souvent		+++	+++	- - -	De nouvelles règles à mettre rapidement en œuvre : Demandes d'aide de la Politique d'Agricole Commune (PAC), règles de discount e-Commerce...
Nombre de règles / Complexité des règles	Elevé	++	+++	- - -	Avantages à long terme de la maintenance. Connaissance métier très pointue : indemnité de départ à la retraite, détection des fraudes ...
	Moyen	++	++	-	
	Bas	-	-	++	
Organisation	Rigide	- -	- - -	+++	Un BRMS peut changer radicalement l'équilibre entre la MOA et la MOE
	Implication forte de la MOA	++	++	-	
Management	Implication forte, porteur de la solution	++	++		
	Opposé, motivation faible	- -	-		
Sentiment de sécurité / confiance		- -	++	++	

En conclusion

Les systèmes experts sont moins à la mode qu'il y a quelques décennies mais leur développement n'a jamais cessé et la puissance de calcul des ordinateurs leur ouvre des perspectives toujours plus vastes.

Cette branche de l'intelligence artificielle offre un fort potentiel dans les secteurs de la banque, des télécoms, des assurances et bien d'autres (instruction de dossier de crédits immobiliers, lutte contre la fraude ou le blanchiment d'argent, recommandation d'offres commerciales, planification des types d'expertise par type de sinistres…).

Le développement de systèmes experts nécessite un type de logiciel spécialisé comme Drools, des savoir-faire nés de l'expérience et de la pratique mais également des compétences en IA.

Nous accompagnons nos clients sur leurs projets innovants pour développer des Systèmes Experts capables de prendre des décisions ou de leur faire des recommandations dans différents domaines.

Nous accompagnons aussi notre partenaire Red Hat dans des missions de consultant sur RH-DM mais aussi en avant-vente sur des PoC (Proof of Concept) et des appels d'offres.

Thierry Guerin
Senior Middleware Architect
Leader BPM (jBPM / Red Hat Process Automation Manager)
Leader Moteur de règles (Drools / Red Hat Decision Manager)
Red Hat Partner
Tel. : +33 (0)6 13 91 67 49
t.guerin@groupeonepoint.com