DMN Modeler Report

DMCommunity Challenge - March 2017

Published: 5/28/2017

By: Edson Tirelli

Table of Contents

Introduction

This is the proposed solution for the DMCommunity challenge from March 2017: Online Dating Decision services.

In this document, you will find the explanation and documentation of the solution. The source code of this model is available here:

- DMN model

This is a standard DMN level 3 solution, and as so, you should beable to execute it using any DMN level 3 compatible engine. In particular, I used Trisotech's DMN Modeler to edit and create the model and Red Hat's Drools to execute it.

Here is a sample code to execute the model in Drools:

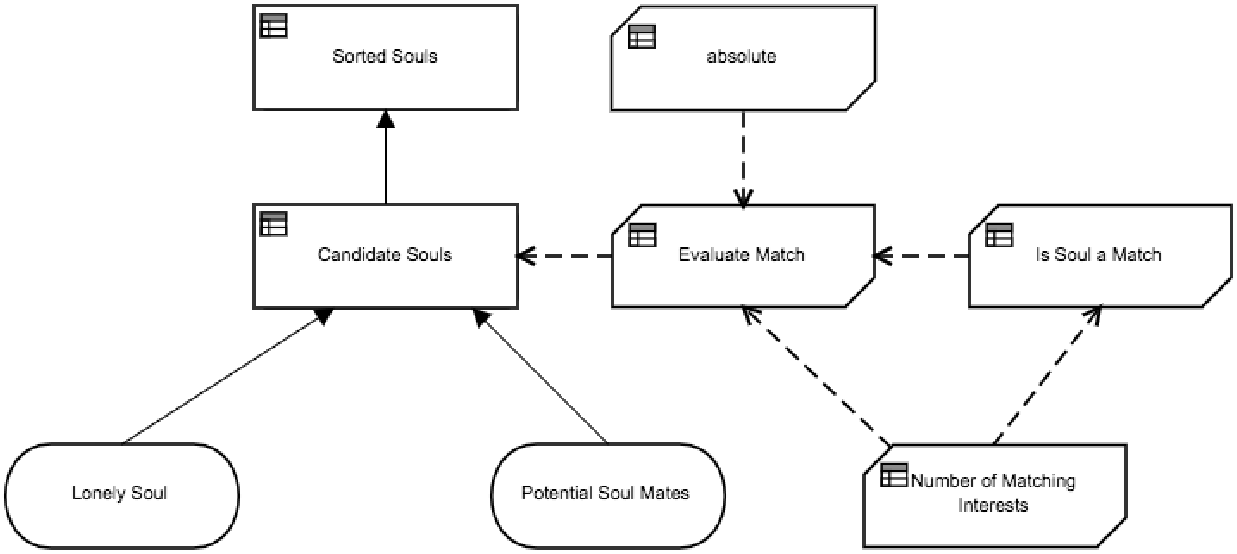
- Drools runner for the model

The runner above, produces the following results:

```
Matches for Bob:
1. Eleonore    - Score =  3
2. Isis        - Score =  2
3. Alice       - Score = -1
4. Grace       - Score = -3
```

DMCommunity Challenge - Match 2017

Decision Requirement Diagram



Elements

Sorted Souls (Decision)

Description

Sorts and returns the the list of matching souls in decreasing score order.

Output Data Type

Type	tCandidates

Decision Logic (Literal Expression)

Sorted Souls
sort(**Candidate Souls**, function(c1, c2) **c1**.Score >= **c2**.Score)

absolute (Business Knowledge Model)

Description

Given a number, this BKM returns the absolute value of that number. I.e., if the number is greater or equal to zero, it returns the number itself. If the number is negative, it returns the number multiplied by -1.

This BKM could easily be implemented with a simple "if" statement, but it is here demonstrating how the model can easily integrate with Java functions to provide functionality not available out of the box in FEEL.

Output Data Type

Type	Number

Decision Logic (Function - Expression)

absolute	
J	(value *Number*)
class	"java.lang.Math"
method signature	"abs(double)"

Candidate Souls (Decision)

Description

Iterates the list of ***Potential Soul Mates*** checking for matches with the ***Lonely Soul.*** Returns a list containing only the matching souls with their corresponding scores.

Output Data Type

Type	tCandidates

Decision Logic (Context)

Candidate Souls	
Candidates *tCandidates*	`for Soul Mate in Potential Soul Mates return Evaluate Match( Lonely Soul, Soul Mate )`
`Candidates[ Is Match = true ]`	

Evaluate Match (Business Knowledge Model)

Description

Evaluates the match between the ***Lonely Soul*** and a ***Candidate***, setting the ***Is Match*** attribute as true if it is a match of false otherwise. It also calculates the ***Score*** for the match.

Requirements are not clear on how to calculate the score, so this decision assumes 1 point for each matching interest and -1 for each year of difference in the ages of the loving birds.

Output Data Type

Type	tCandidate

Decision Logic (Function - Context)

Evaluate Match

(Lonely Soul *tProfile* , Candidate *tProfile*)

Profile1 *tProfile*	`Lonely Soul`
Profile2 *tProfile*	`Candidate`
Is Match *Boolean*	`Is Soul a Match(Lonely Soul, Candidate) and Is Soul a Match(Candidate, Lonely Soul)`
Score *Number*	`Number of Matching Interests(Lonely Soul, Candidate) - absolute( Lonely Soul.Age - Candidate.Age )`

Is Soul a Match (Business Knowledge Model)

Description

Returns true if the candidate is a match for the given lonely soul. According to the requirements, a candidate is a match if and only if:

1. Gender of the other person must be one of the acceptable genders
2. Age of the other person must be within the acceptable range
3. City must match exactly
4. Matching interests of the other person must match at least the number specified

Output Data Type

Type	Boolean

Decision Logic (Function - Expression)

Is Soul a Match	
F	(Lonely Soul *tProfile* , Candidate *tProfile*)

```
list contains(Lonely Soul.Acceptable Genders, Candidate.Gender)
and
Candidate.Age between
    Lonely Soul.Minimum Acceptable Age and
    Lonely Soul.Maximum Acceptable Age
and
Candidate.City = Lonely Soul.City
and
Number of Matching Interests(Lonely Soul, Candidate) >= Lonely Soul.Minimum
Matching Interests
```

Potential Soul Mates (Input Data)

Description

A list of profiles of the potential soul mates.

Input Data Type

Type	tProfiles

Lonely Soul (Input Data)

Description

The profile of the user for which potential soul mates are being looked for.

Input Data Type

Type	tProfile

Number of Matching Interests (Business Knowledge Model)

Description

Returns the number of matching interests between the ***Lonely Soul*** and the ***Candidate Soul Mate***.

Output Data Type

Type	Number

Decision Logic (Function - Context)

Number of Matching Interests	
(Lonely Soul *tProfile* , Candidate Soul Mate *tProfile*)	
Matching Interests *tBooleans*	`for Interest in Lonely Soul.List of Interests return list contains(Candidate Soul Mate.List of Interests, Interest)`
`count( Matching Interests[item = true] )`	

Data Types

tProfile

Name	Text
Gender	tGender *"Male", "Female"*
City	Text
Age	Number
List of Interests	tInterests
Minimum Acceptable Age	Number
Maximum Acceptable Age	Number

Acceptable Genders	tGenders
Minimum Matching Interests	Number

tGender

Text *"Male", "Female"*

tGenders

tGender *"Male", "Female"*

tInterests

Text

tProfiles

tProfile

tCandidate

Profile1	tProfile
Profile2	tProfile
Is Match	Boolean
Score	Number

tCandidates

tCandidate

tBooleans

Boolean