The Effectiveness of DMN Portability

Mark Proctor[1][0000-0002-1746-072X], Edson Tirelli[1], Davide Sottara[2], Bruce Silver[3], Jacob Feldman[4], Mélanie Gauthier[5]

[1] Red Hat
[2] Department of Biomedical Informatics, Arizona State University, Tempe, USA
[3] Method & Style
[4] OpenRules
[5] Trisotech

Abstract. Decision Model and Notation (DMN) is an OMG standard to ensure that decision models can be shared between multiple stakeholders, and/or created, analyzed and executed on platforms provided by multiple vendors. DMN is already showing promise, with several industrial business rule vendors providing support. This compares well to previous standards like Rule Interchange Format (RIF) and Production Rule Representation (PRR) which failed to gain industry adoption with the business rules community. This paper introduces DMN with a focus on the portability of its executable profile across computational platforms. The paper proposes a cross vendor collaboration challenge, "Vacation Days" from the Decision Management Community, to test the effectiveness of the standard for use in cross vendor situations. The different degrees of compliance and extensions as supported by the vendors is also discussed.

Keywords: Decision Model Notation, Object Management Group, Rules, Decision Tables, Decision, Business Rules.

1 Introduction and Background

When looking at standards for business rules, of those proposed, none have yet gained mature adoption with the main "business rule" vendors - such as Oracle, IBM and Red Hat. The most notable efforts to date have been Rule Interchange Format (RIF), Production Rule Representation (PRR) and the Rule Markup Language (RuleML). In the case of RIF those vendors focused around the Production Rule Dialect (RIF-PRD). Both RIF-PRD, PRR and Reaction RuleML [5] focused on the interchange between production rule systems, which are at the basis of most vendor solutions. Multiple business rule vendors were involved in the development of those standards, but neither standard, to date, had any success in the community, resulting in lack of adoption. This presents a challenge to end users who worry about vendor lock-in and portability, with costly custom migration efforts if they decide to move between

vendors. Nevertheless, lack of support from the vendors is a disincentive for end users, and lack of demand from end users is a disincentive for vendors, resulting in vicious circles.

RIF was a large effort, from W3C, that attempted to provide interchange between a number of different rule types using XML. RIF rule semantics were modelled on Horn Logic [6] and used the concept of dialects, where it provided the CORE dialect and others such as Base Logic Dialect (BLD) and Production Rules Dialect (PRD) could be layered. Five dialects were proposed as part of the standard. The MISMO organization presented a proof of concept "loan example" to test cross vendor interoperability using ILOG JRules (now IBM ODM) and Red Hat Drools using the RIF PRD dialect. Despite the success of the demonstration, and the involvement of both vendors, there was no further progress in adoption of this standard.

PRR was a more focused effort, from OMG, that attempted to only provide a standard for production rules with a visual model representation (UML) as well as an XML document. It proposed two meta models, PRR-Core and PRR-OCL, with the latter committing to the use of BasicOCL, a subset of the OCL standard, to express the rule logic, while the former did not make any specific recommendation as to what expression language should be adopted.

RuleML, along the lines of RIF, focuses on providing a concrete syntax for the expression of a variety of types of rules, with varying degrees of expressivity according to the underlying logic(s). To this end, RuleML is an actually a modular family of languages, which can be composed to target the exact level of expressivity required by each use case.

In contrast to previous efforts centered on business (production and/or ECA) rules, DMN does not aim to be an interchange language between systems or offer full production rule semantics. DMN allows to express the specifications of how a business problem should be solved by means of processing information according to a given corpus of knowledge (either declarative or imperative, prescriptive or descriptive), with the aim that different "reasoning agents", human or computer, can execute it. Since software agents executing production rules fall under this category, DMN provides an opportunity for business rule management system vendors to specify modular knowledge bases, and map them to something that can be executed on their production rule systems. The standard itself defines the required semantics of the standardized decision modeling constructs, but allows different vendors to execute them in a number of different ways. In fact, DMN focuses on the ability to define decisions, their information requirements, and the knowledge that influences the decision making processes. The assumption is that the results of one decision (making process) provides the answer to a precise business question, that can also be used as an input to other decision(s) until all the relevant answers are derived. The dependencies between decisions can be conceptualized as a graph, and expressed by means of a "Decision Requirements Graph". The actual decision making logic is abstracted by the standard, and, when explicitly expressed, is expected to be formalized as a computable "expression" in some kind of language and notation, the most common being a Decision Table.

The purpose of this work is to test the effectiveness of this standard, using the "Vacation Days" example taken from the DMCommunity Jan 2016 challenge, by executing a cross vendor workflow defined so that multiple vendors are involved in the authoring, analysis and runtime execution.

The life cycle of a knowledge artifact (the expression of a piece of business knowledge expressed in some knowledge representation and reasoning language) may include activities such as the authoring, curation, storage and retrieval, verification, simulation, execution, revision and presentation. Different vended systems are likely to provide different subsets of capabilities around such activities: the goal of DMN is to support full interoperability between such systems, and this endeavor aims to assess how far away a number of mainstream concrete implementations are from this ideal state.

2 Decision Model and Notation Overview

The Decision Model and Notation (DMN) [1] is a standard by the Object Management Group (OMG) [2]. It aims to bridge the gap between business models and executable implementations by defining a high level language with a common notation that is readily understandable by users across the business, from analysts to developers.

The actual specification builds on the experience acquired defining the Business Process Model and Notation (BPMN), its sibling specification. BPMN is used to model Business Processes. Interoperability and seamless integration between BPMN and DMN is also a goal of the specification.

DMN has been designed for "naturalistic" decision models, which mainly focus on the description of how humans make decisions, and "computable" decision models, which are intended for automated execution e.g. by a rule engine. From the perspective of the latter, DMN defines primarily 5 concepts:

1. Decision Requirement Diagrams (to establish relationships between data, knowledge and decisions)
2. Decision Tables (to specify decision making logic)
3. FEEL (the formal expression language - for information processing within a decision making process)
4. Boxed expressions (a graphical format for complex expressions)
5. Metamodel and schema (formal definition of model elements, and XML interchange format)

The specification defines 3 incremental levels of conformance. For the formal definition of each conformance level, please refer to the DMN 1.1 [1] specification, page 9. Here is an informal definition of the compliance levels:

1. Conformance Level 1: A conformance level 1 implementation is required to support Decision Requirement Diagrams, Decision Logic and Decision Tables, but the model does not need to be executable. Any language can be used to define the expressions, including natural, unstructured languages.

2. Conformance Level 2: A conformance level 2 implementation is required to support the same as Conformance Level 1, but expressions should be written using the S-FEEL (Simplified FEEL) language, as defined in chapter 9 of the specification. Conformance level 2 models must be fully executable.
3. Conformance Level 3: A conformance level 3 implementation is required to support everything in Conformance Level 2, plus the full set of boxed expressions. In addition to that, all expressions can be written using the full FEEL language, as defined in chapter 10 of the specification.

DMN was designed for BPMN2 interoperability (see **Fig. 1**) and decision models can be linked from a decision task within a business process model expressed using the BPMN2 standard. This relationship is usually manifested allowing to navigate from a business process diagram (that renders a BMPN2 process model) to a decision model diagram (that renders a DMN decision model).

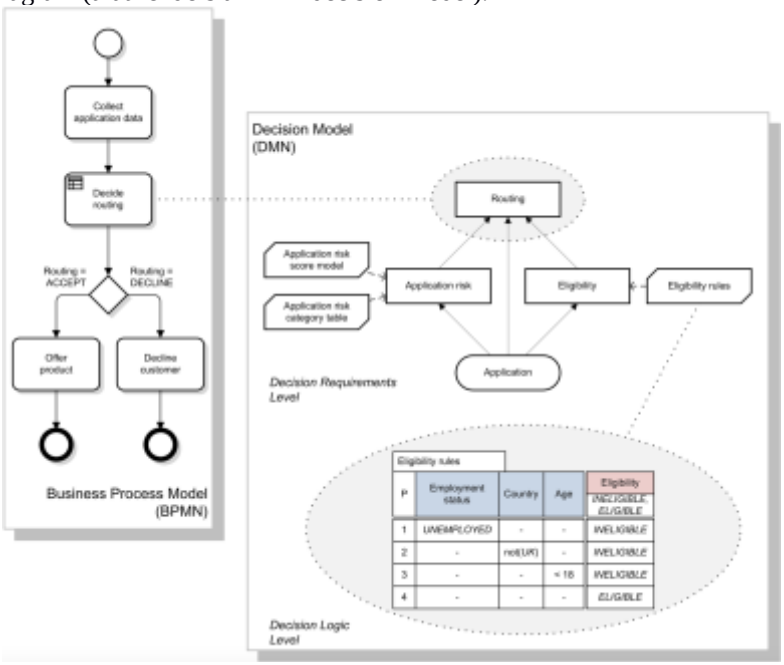

***Fig. 1** BPMN2 Diagram using a DMN node [1]*

2.1 Decision Requirement Graphs

A Decision Requirements Graph (DRG) (see **Fig. 2**) is a directed acyclic graph of nodes (see **Fig. 3**) and the relationship between those nodes (see **Fig. 4**), that models the decision making logic in a given business domain. A Decision Requirements Diagram (DRD) is a "view" on a DRG, i.e. a subset of the nodes and relationships, that emphasizes the decision making logic from the full DRG.

These graphs model the chain of decisions and its dependencies, with each decision node receiving an input and producing outputs for other decision nodes to

consume. There is no single starting or end point. Instead each node defines the variables it needs in order for it to be able to evaluate: whenever those variables are available, an (informed) decision can be evaluated to produce more information, which in turn can result in other nodes being satisfied and evaluated. Nevertheless, DMN does not dictate whether decision models should be evaluated in a 'forward chaining' rather than a 'backward chaining' or a 'hybrid chaining' manner. The orchestration of decisions is left to the implementation (engines), or by orchestrating the decisions by means of business processes.

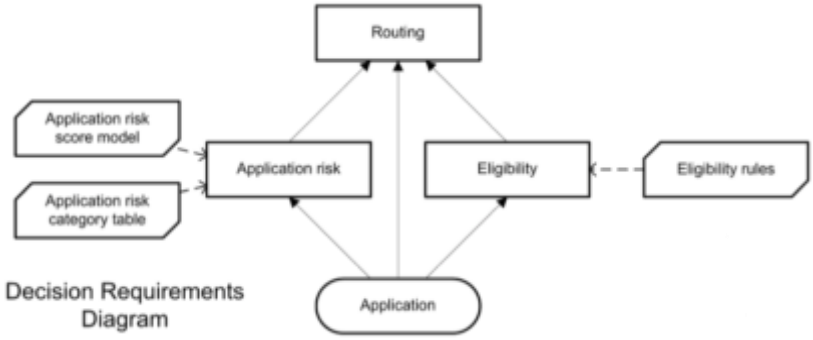

Fig. 2 *Decision Requirement Diagram [1]*

Component		Description	Notation
Elements	Decision	A decision denotes the act of determining an output from a number of inputs, using decision logic which may reference one or more business knowledge models.	Decision
	Business Knowledge Model	A business knowledge model denotes a function encapsulating business knowledge, e.g., as business rules, a decision table, or an analytic model.	Business knowledge
	Input Data	An input data element denotes information used as an input by one or more decisions. When enclosed within a knowledge model, it denotes the parameters to the knowledge model.	Input data
	Knowledge Source	A knowledge source denotes an authority for a business knowledge model or decision.	Knowledge source

Fig. 3 *DRD Component Elements [1]*

Component		Description	Notation
Requirements	Information Requirement	An information requirement denotes input data or a decision output being used as one of the inputs of a decision.	
	Knowledge Requirement	A knowledge requirement denotes the invocation of a business knowledge model.	
	Authority Requirement	An authority requirement denotes the dependence of a DRD element on another DRD element that acts as a source of guidance or knowledge.	

Fig. 4 *DRD Component Requirements [1]*

2.2 Decision Tables

DMN supports decision tables as one form of expression to determine the outcome of a decision making process. DMN Decision tables support both vertical and horizontal layout with single or multiple output columns, as well as a crosstab layout. Decision tables also specify a hit policy which controls the execution behavior of the table. **Fig. 5** shows a horizontal layout, with rules as rows and a single result column.

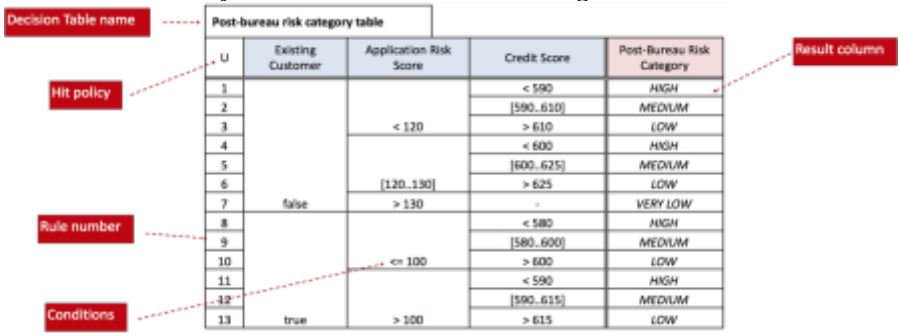

Fig. 5 *Decision Table [1]*

DMN Hit Policies

Single Hit

- **U**nique - Only one rule may match
- **A**ny - Multiple rules may match but all give the same output value
- **P**riority - The output with the highest priority is selected, where priority is given by the listed order of allowed values in the output column heading.
- **F**irst - Select the output of the first rule that matches.

Multiple Hit

- **O**utput Order - Returns all hits in decreasing output priority order.
- **R**ule order - Returns all hits in rule order.
- **C**ollection - returns all hits in arbitrary order. An operator (‘+’, ‘<’, ‘>’, ‘#’) can be added to apply a simple function to the outputs. If no operator is present, the result is the list of all the output entries.

2.3 FEEL

The DMN standard defines a new expression language called FEEL, that stands for "Friendly Enough Expression Language". The purpose of this new language is to define standard execution semantics that would be common to all models, allowing the model to produce the same results independently from the runtime environment.

It is a language designed to appeal to non-technical users, and the goal was that any user capable of authoring typical spreadsheet formulas would be able to learn and use FEEL.

In FEEL, every expression is a formula that produces a value. FEEL expressions are side effect free and variables are immutable.

FEEL defines just a few common data types: string, boolean, number, date, time, and duration types. These types can be restricted to enumerated values or ranges, and can be combined to form complex data structures. FEEL variables may include lists and tables, as well. FEEL provides a large number of built-in functions and operators.

2.4 Boxed Expressions

Boxed expressions are expressions presented in tabular formats. It is a way to decompose complex logic in tables making them more readable. For instance, the following example (see **Fig. 6**) is a boxed expression called "Context". A Context is a list of key-value pairs represented as a table with two columns, where the first column is the key (or variable name) and the second column is the value (represented as a FEEL expression). The last row of a Context Boxed Expression does not have a name and the result of its value expression is the result of the boxed expression as a whole.

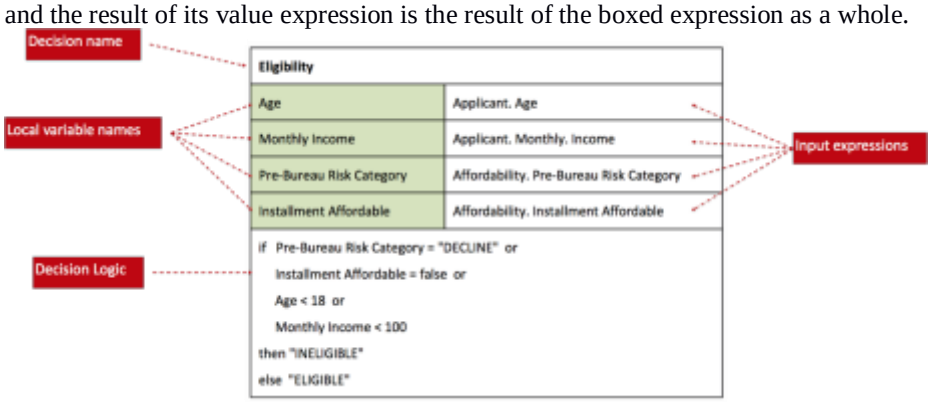

***Fig. 6** Boxed Expression [1]*

DMN supports the following types of boxed expressions:

- Decision Tables
- FEEL expressions
- Boxed invocation
- Boxed context
- Boxed lists
- Relations
- Boxed functions

Boxed expressions can be nested. For instance, a Decision Table can be a value of a context entry in a Context Boxed Expression.

2.5 Metamodel and XML Schema

The metamodel defines all the components of a DMN model, its properties and semantics. By bringing formal precision to the element definitions, they ensure DMN models can produce the same results when executed by different decision engines.

The XML schema, on the other hand, is the standard syntax that enables interoperability between different engines. One of the DMN metamodels, the "decision metamodel", is shown in **Fig. 7**. DMN defines XML for each its metamodels.

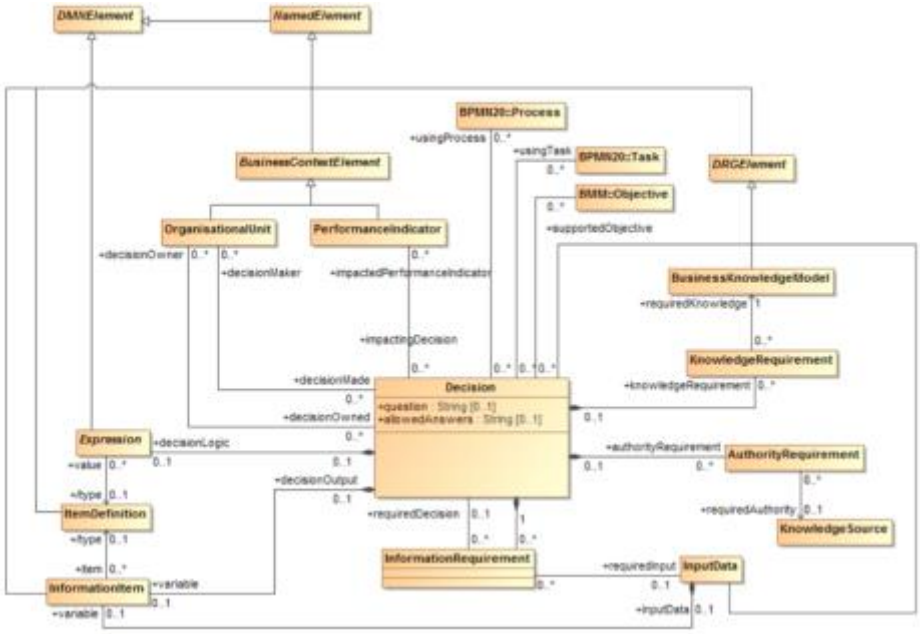

***Fig. 7** Decision metamodel*

3 Degrees of Vendor Support

While DMN has gained good vendor support for a common denominator, already allowing for end users to work in a multi-vendor situation, it still has to gain wider support for full compliance. Known areas that have limited support include:

- Full FEEL support, including variable names with spaces and special characters
- Boxed Expressions
- Contexts
- Loops
- Filter Expressions (table lookups)
- Decision Requirement Diagram

As of now there is no clarity or transparency on what is supported across vendors, or what this common denominator is. Most vendors have not yet gained full CL2, so there is a long way to go before wider CL3 adoption is made. This places a burden on the end user to understand what this common denominator is and keep their models within that to ensure what they create is portable. Work is underway on a community driven TCK [8] that over time may help bring clarity for end users. This paper is not going to detail what each vendor does or doesn't support, this must be done fairly and would require a full TCK with a feature by feature breakdown which is not available until the TCK reaches more maturity.

Furthermore, several vendors are proposing extensions to address customer needs. This will also represent problems with portability and end users should be fully aware of those extensions, before adopting them. In some cases, efforts may be underway to have those extensions adopted within the standard.

4 The Vacation Days Challenge

4.1 The Problem

The Decision Management Community (http://DMCommunity.org) runs regular challenges which provide plenty of potential source material. Most vendors do not have full DMN support yet so it's important that we pick an example that uses the common denominator that is supported. Vacation Days [9] challenge was submitted for January 2016 and multiple vendors provided DMN based solutions that fell within the CL2 conformance level. It was a nice challenge with elegant solutions, although a little on the small and simple side, that broke down into multiple different concerns that helped create a clean solution that demonstrates both DRDs and decision tables. For this paper the solution submitted by Jacob Feldman [12], from OpenRules, was used as the base. Small changes where applied, to ensure it could run across all vendors' products. This was not as easy as first hoped. For example, 'if' was not supported by all vendors, which was used in the solution submitted by Gary Hallmark [11], from Oracle. Professor Jan Vanthienen provides a scoring metric [10] for all submissions and evaluates them on the following criteria: traceable, maintainable, overview and conformant.

The Vacation Days Challenge

The number of vacation days depends on age and years of service. Every employee receives at least 22 days. Additional days are provided according to the following criteria:

1. Only employees younger than 18 or at least 60 years, or employees with at least 30 years of service will receive 5 extra days.
2. Employees with at least 30 years of service and also employees of age 60 or more, receive 3 extra days, on top of possible additional days already given.
3. If an employee has at least 15 but less than 30 years of service, 2 extra days are given. These 2 days are also provided for employees of age 45 or more. These 2 extra days cannot be combined with the 5 extra days.

4.2 The Solution

Following the DMN standard, the high-level solution is modelled in a DRD (Decision Requirements Diagram) that is presented in **Fig. 8**. Each node of the diagram is explained in the following sections.

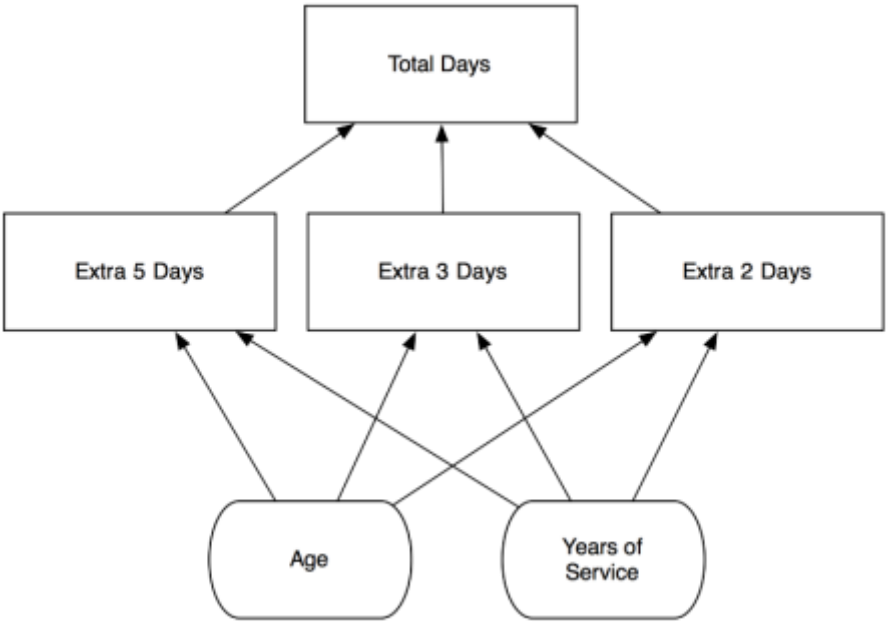

***Fig. 8** DRD for Vacation Days*

Input Nodes

This problem statement defines two inputs: the age and the number of years of service of the employee.

Extra Days

The problem statement presents 3 different rules/cases for extra vacation days. Although the cases can be combined into a single expression or single decision table, in order to improve the maintainability of the solution, a separate decision table was defined for each case. The three cases are presented in figures **Fig. 9**, **Fig. 10**, **Fig. 11** and **Fig. 12**

Extra 5 Days

F	Age	Years of Service	Extra 5 Days
1	<18	-	true
2	>= 60	-	true
2	-	>=30	true
3	-	-	false

Fig. 9 *Extra 5 Days*

Extra 3 Days

F	Age	Years of Service	Extra 3 Days
1	-	>=30	true
2	>=60	-	true
3	-	-	false

Fig. 10 *Extra 3 Days*

Extra 2 Days

F	Age	Years of Service	Extra 2 Days
1	-	[15..30)	true
2	>=45	-	true
3	-	-	false

Fig. 11 *Extra 2 Days*

It is important to note that for simple decision tables like this, there are several hit policies that could be used with the same result (for instance, Priority ("P") policy would also work).

4.3 Total Vacation Days

Calculating the total vacation days is then a trivial summation of all the component decisions, with a single caveat: the problem statement indicates that the extra vacation days from cases 1 and 3 are not cumulative. There are several ways of doing this and the example uses a decision table, because it is easy to understand and works across all the tools:

Total Vacation Days

C+	Extra 5 Days	Extra 3 Days	Extra 2 Days	Total Vacation Days
1	-	-	-	22
2	true	-	-	5
3	-	true	-	3
4	false	-	true	2

Fig. 12 *Total Vacation Days*

The "Collect SUM" hit policy adds up the values of all matching rows, calculating the total vacation days. The model is very easy to read and understand.

4.4 Results

Here is a table with some results generated by the solution.

Input		Output
Age	Years of Service	Total Vacation Days
16	1	27
25	5	22
44	20	24
44	30	30
50	20	24
50	30	30
60	20	30

Fig. 13 *Results*

4.5 Cross Vendor Collaboration Workflow

The cross vendor collaborative workflow involves 5 different vendors, each addressing a different part of the workflow.

- Authoring: Signavio, Trisotech
- Execution: OpenRules, Drools
- Analysis: Method&Style

The demo was executed as shown in **Fig. 14**.

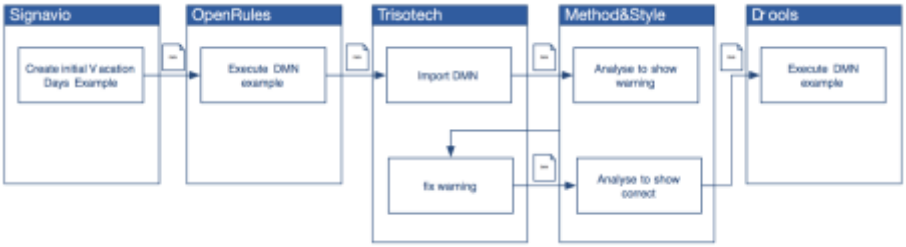

Fig. 14 *Cross Vendor Collaboration Workflow*

5 Conclusion

Using a subset of DMN we have managed to take an example involving DRDs and decision tables and create, analyze and execute those across multiple vendor's products. While there is little full CL2 or CL3 compliance yet, this is a stronger starting point that either RIF-PRD or PRR managed to attain. Although it should be noted that DMN's goals are different. RIF-PRD and PRR were about interchange between two systems and not the primary authoring target of those systems. DMN primarily addresses portability, allowing the same document to run in different systems – rather than interchange native documents between systems. DMN still presents many challenges to early adopters due to lack of clarity on the degree of support across vendors and this will be the next challenge for DMN. As shown in the demo it can be a case of trial and error to find that portable common denominator. This makes it difficult to know what can or cannot be used from DMN while still ensuring models remain portable. The community driven TCK will be an important tool to help drive DMN to that next level, providing the clarity and transparency that customers need. In the early stages where there is still not full CL2 or CL3 compliance, it will not be enough to say whether a vendor meets CL2 or CL3. Instead it must provide a detailed breakdown of what is and isn't supported by a vendor, so end users can use this information to guide them through the trial and error process of making portable documents.

References

1. Decision Model Notation 1.1, http://www.omg.org/spec/DMN/1.1
2. Object Management Group, http://www.omg.org/
3. Rule Interchange Format, https://www.w3.org/TR/rif-overview/
4. Production Rule Representation, http://www.omg.org/spec/PRR/

5. Reaction RuleML, http://wiki.ruleml.org/index.php/Specification_of_Reaction_RuleML_1.0
6. Rule Interchange Format Working Group Charter, https://www.w3.org/2005/rules/wg/charter#horn
7. Bost T., Bonnard P., Proctor M. (2007) Implementation of Production Rules for a RIF Dialect: A MISMO Proof-of-Concept for Loan Rates. In: Paschke A., Biletskiy Y. (eds) Advances in Rule Interchange and Applications. RuleML 2007. Lecture Notes in Computer Science, vol 4824. Springer, Berlin, Heidelberg
8. Community Driven TCK, https://agilepro.github.io/dmn-tck/index.html
9. Vacation Days Challenge, 1. https://dmcommunity.org/challenge/challenge-jan-2016/
10. Challenge Scores, https://dmcommunity.files.wordpress.com/2016/02/dmc-challenge-jan2016.pdf
11. Gary Hallmark Vacation Days submission, https://dmcommunity.files.wordpress.com/2016/01/good-decision-table-challenge-jan-2016-garyhallmark.pdf
12. Jacob Feldman Vacation Days submission, https://openrules.wordpress.com/2016/01/04/decision-table-for-vacation-days-calculation/