Decision Management Community

Challenge Jan-2016

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 can not be combined with the 5 extra days.

INTRO

I was happy to see a large number of submissions to the challenge. Just to make things clear, I did not start the challenge and I did not pick the example, so don't blame me if you did not like the case. But I am happy to give some feedback, as requested. Thanks for all the extra work, DMC!

The comments *are by no means an indication of the capabilities/weaknesses of specific tools or submitters*. They are just a feedback to the courageous decision table modelers who tried to guess my preference and help to build support for good decision table and DMN practices. Many thanks to all of you.

CRITERIA

The example was a real one (some years ago). I sometimes used it in presentations, so I can imagine some of you have seen a decision table form of it earlier. But that should be no reason not to look for the best solution.

Of course there is no best solution for all purposes. The holiday rules show how decision modeling involves:

- modeling the set of decisions and rules
- providing an overview for business purposes (easy to analyze)
- looking for inconsistencies or omissions (easy to validate)
- traceability to the knowledge authority
- ease of maintenance
- efficient implementation

- easy to build
- flexible reasoning
- etc.

So it should be no surprise that many submissions have various advantages.

There are also some specific considerations about this example:

- Additional days (5, 3, 2) are only added once (that is an assumption indeed), but some of them can be combined.
- The 2 extra days cannot be combined with the 5 extra days.
- You can not have more years of service than your age – the legal minimum age (child labour is illegal).

And there are some business questions that should be answered easily:

- Decision making: John Doe is 50 and has 30 years of service, how much does he get (30 days)?
- Overview: Who finally gets the 2 extra days?
- Maintenance: What if the combination policy for 2 and 5 bonus days changes?
- Analysis: Are there some strange assignments in giving additional days? Yes, if you turn 18, e.g., you loose 5 bonus days.

FIRST CONCLUSIONS

One of the things worth mentioning is that there is a large variety of tools and formats. Now that DMN 1.1 offers a standard representation, it would be good for exchange and readability if the common notation is used:

- [45..60) or [45..60[are well-defined notations and much more compact than "(Age >= 45) and (Age < 60)"
- Decision tables can return the outcome of one rule (the first, the only one, whatever) or the outcomes of all matching rules. It is not always clear from the submissions what is meant. That is exactly the purpose of the hit indicator: tell us how to read the table. If the hit indicator is not used, only the outcome(s) of one rule will be returned.
- An irrelevant input entry is indicated with "-", not '*' or blank.

AN OVERVIEW OF THE SUBMISSIONS

Let us have a look at the solutions, one by one, in the order presented on the community site.

OpenRules 1 – Unique-Hit Decision Table

A typical nice straightforward table. It is complete and consistent. John Doe gets his 30 days and it is easy to see who gets 5 extra days, by leaving these numbers unadded in the outcome cells. Rows 5 and 6 could be contracted.

It is clear who gets 2 bonus days. But what was the original rule for getting 2 days? Is it *lost in translation*?

OpenRules 2 – Multiple-Hit Decision Table

A lot of work has to go into this table. First the OR-ed rules are enumerated, where we have to make sure that the 5 and 3 days are only added once, and that the 2 extra days are never added on top of the 5 by carefully reformulating these rules. Because an OR in decision tables leads to multiple rules and in this case we want to make sure that the rules for 5 (or 2 or 3) do not overlap. Otherwise the same number of extra days would be given more than once. Tooling can overcome this difficulty, but is it really close to the problem formulation? What if the business decides that the 5 and 3 days can not be combined anymore. Who gets both anyway? Very error-prone indeed.

OpenRules 3 – Distinct decision tables

Similar to Gary's solution, the global table nicely solves the combination problem (what can be combined?). The separate tables are close to the specification. But are these first-hit tables? I don't see any indication of this hit policy. The last row is the default value. What if I would put the last row first? This is clearly a default row (called the ELSE-row).

The tables are very traceable and maintainable. But how can I (poor human with a short memory) find John Doe in an instant? I would have to answer a question about his age in the 5-table, then again in the 3-table, etc.

DMN – Distinct decision tables

Modeling serves a purpose. Instead of imprecise natural language, a more formal specification of the problem statement takes away ambiguities and confusion. DMN allows to model this decision and its constituting components and Gary shows us how to do this elegantly in DMN. The bonus rules are listed in distinct tables, and the boxed expression for the top decision tells us how to combine the three bonuses. This is very flexible, traceable and maintainable.

Have a look at some DMN conventions:

- A boxed expression to set the base days.
- The A (Any) hit-indicator to list rules connected by "or", with the same outcome. This allows to stay close to he original text (and automatically avoids giving the 5 days more than once).
- The underlined value false to state the default outcome.
- The use of "," in '<18, >=60' to indicate 'or'.
- The expression to calculate the total days (it could also be a table, for people who do not understand 'if then else').

The model is of course fully executable.

John Doe gets his 30 days. It is only a little difficult to spot the difference between John with 29 and John with 30 years of experience. How much can he gain by staying an extra year? 6 days.

Corticon 1 – Multiple-hit table

The original rules are recognizable in the table and in order to avoid double bonuses (5+5), the rules are adapted. The design process is well illustrated. This is a very good analysis of how to move from the original rules to the clean multiple-hit table. And it was even noticed that some employees

receive no bonus and how to solve that. The result clearly distinguishes the various bonuses, but somehow it still hides how 2, 3 and 5 days can be combined, because it is implicit in the rules, not explicit. How do I know that the bonus days have to be added?

Corticon 2 – Separating Eligibility and Exclusion

This is a nice set of tables. The combination rules are now separated from the eligibility rules and can be managed separately. The tables are multiple-hit, every result has its own set of rules, and some results can be combined.

Adding flexibility: This is now turning into a powerful and flexible mechanism. Overlap in eligibility can be shown, although a unique-hit table might have shown a stronger picture here. Sometime it is a little hard to figure out if one rule should be chosen, or all the rules that apply.

Corticon 3 – Unique-hit table with business view

The distinction between business and implementation view is nice. The major message here, however, is that now a unique, single-hit table is used. This indicates that something is still useful to add to the multiple-hit tables of the previous solutions: the overview.

Trisotech 1 – Unique decision table

Complete, consistent and easy to overview the result. Nice to refer to the original rules for traceability. Notice the world 'rules' here and in the solution, as opposed to rows, so a distinction is made, which is a plus.

What if the original combination rules for 5, 3, 2 would change? Then the table has to be rebuilt indeed, which is a little complex.

Rewriting the original text is a plus, but I wouldn't have used a 'first rule that applies' approach here.

Trisotech 2 – First-hit table

As you indicate, I do not like such a first-hit table. Who finally gets the 22+2 days? Everyone >=45 with >=15 years of service? No, only if they did not receive more days in the previous rows. And who was that again? Humans can not perform such complex logical operations on a Friday evening. Hard to validate, although in this case maintenance is not such a problem, because the rows are ordered according to the original rules.

Avola – Any-hit table

The 'Any' table allows to stay rather close to the initial specification, keeping the "OR" in "at least age 60 or more than 30 years of service". But the rules have to be reformulated anyway to avoid 2 bonus days on top of 5. Why not go one step further and avoid the overlap in the last 2 rows? The last row, e.g. could show [18..60).

I would also use DMN range and hit indicators.

Blueriq 1 – Unique-hit table

Fine, although I would stick to the standard '-' for irrelevant and [18..45) or [18..45[for the range of values. The '[]' for otherwise can be useful, but not in this simple case. It might also be confused with an empty range. Better without the '[]' indeed. The individual bonuses have disappeared by adding them.

The version with "UNKNOWN": Unknown values can be very important for correct and smart decision making. But is this not rather part of the execution environment? Is the decision logic really different if input values are unknown?

Blueriq 2 – Unique-hit table for additional days only

Taking out the base days can indeed be a nice way to deal with the default of 22. But adding the bonus days together is a little hard for traceability and maintenance.

Blueriq 3 – Decision table with the most frequent outcome as default

This table is compact, but has the same disadvantages as a table with an ELSE-column: hard to have a good overview and analyze the situation.

Blueriq 4 – A table with complex conditions

Nice if a tool can do this, but I gave up trying to figure out if the table is even correct. Difficult to understand, as you mention.

Blueriq 5 – The smallest decision table

Compactness is always an advantage if it doesn't harm other criteria. But it does here. Difficult for business, as you mention.

RapidGen 1 – First-hit table

This is actually an "any"-hit table with an ELSE-column. The rules only overlap if they lead to the same outcome ("or"), e.g. rules 2 and 3. The initial action is a nice way to indicate the default.

Hard to figure out if the table is complete. Of course it is complete by definition because of the ELSE-column, but it is difficult to see that this only contains the case Age [18..45) and Service < 15.

Splitting the input ranges into Limited-entry (Y, N) inputs saves space, but is always harder to understand.

RapidGen 2 – Unique-hit table

The rules do not overlap so this would be a unique-hit table without the ELSE-column. Actually, the ELSE-column does not cover any cases, so it can be omitted. Splitting the input ranges into Limited-entry (Y, N) inputs is still harder to understand.

Dealing with invalid input data and calculated years is a nice addition. I would keep it out of the decision table though. Nice implementation.

IBM ODM – Multiple-hit table

Some assumptions have to be made indeed. The 5 days are only given once. Also the 2 days, but how could you know?

The bonus is hardcoded, so that is a little difficult to maintain.

The table has some nice properties in that all the rows (except the first one) are non-overlapping. At least if you read <45 as [18..45). But the order is a little strange and harder to validate manually. Is that the reason why I don't find an outcome for Age [45..60) with Service < 15? Or do you assume that a 45 year old always has more than 15 years of service? That is a little too fast. And what about Age [18..45) with Service < 15?

Rule #2 of the text uses the words: "Employees with Service >= 30 *and also* Employees with Age >= 60…". Isn't this a common ways to express "*or*" in daily language? So a 45 year old with 30 years of service should receive a bonus of 8.

The first row always applies, therefore this is a multiple-hit table (although the + is missing). But the rest of the table is first-hit or even unique-hit. I would not mix both forms.

FlexRule – Distinct decision tables

Similar to Gary's and Jacob's solution with the combinations in the Total table (with similar advantages). But how do I read this total table? Can I stop after a hit? No, I have to continue, so it is multiple-hit (where outcomes have to be added). In the other tables, I have to stop after a hit, or everything turns false, so these are first-hit tables. Where do I see this? I would indicate the distinction between these types of tables and show the hit-indicator.

Finding John Doe is not an easy task.

EVALUATION

There are different types of solutions and they reflect where we put the focus in this decision challenge:

- **Building a model which clearly captures the original specification and can easily be traced back to it.** This is flexible and easy to maintain. If the bonus rules or the combination rules change, the model easily follows. For good examples, have a look at Gary's DMN solution. OpenRules 3, Corticon 2 and FlexRule are similar.
- **Buiding a model which easily shows the overview, allowing to analyze and validate the business concerns.** These models allow to compare the bonus days over age categories, spot strange outcomes and show the result in a blink of the eye. For some good examples, see Trisotech 1, Corticon 3, Open Rules 1 and RapidGen1.
- **Building a model (and tables) that tries to combine both views: show the final outcomes and ensure traceability to some extent.** That is often a compromise and challenging. See Avola, Blueriq, IBM ODM.

Note that all this is possible in DMN, in a standardized way, and immediately executable from the model.

THE OVERVIEW

	Traceable	Maintainable	Overview	DMN conformant	Score
OpenRules 1:	medium	medium	high	high	*very good*
2:	medium	medium	medium	high	*good*
3:	high	high	medium	medium	*excellent*
DMN:	high	high	medium	high	*excellent*
Corticon 1:	medium	low	medium	medium	*good*
2:	high	high	medium	high	*excellent*
3:	medium	medium	high	high	*very good*
Trisotech 1:	high	medium	high	high	*excellent*
2:	high	medium	medium	high	*very good*
Avola:	high	medium	medium	medium	*good*
Blueriq 1:	medium	medium	high	medium	*good*
2:	medium	medium	high	medium	*good*
3:	medium	medium	medium	medium	*good*
4:	medium	low	medium	low	*low*
5:	medium	low	low	low	*low*
RapidGen 1:	high	medium	medium	medium	*good*
2:	medium	medium	high	medium	*very good*
IBM ODM:	medium	medium	medium	medium	*good*
FlexRule:	high	high	medium	low	*good*

Final note 1: With all these possibilities, it is important to understand and exchange each other's models. So, follow the DMN conventions.

Final note 2: Trying to satisfy all objectives in one model is not an easy task. Here is a challenge for the next generation of DMN tools: model transformation, in order to deal with multiple objectives.

Jan