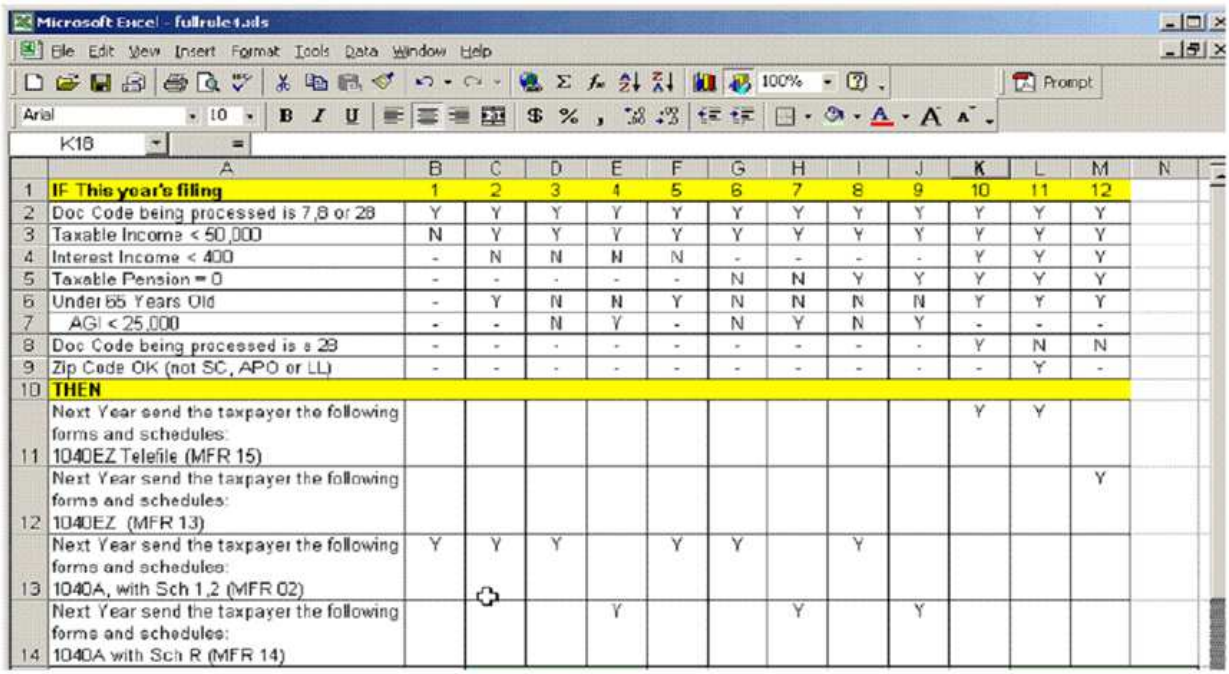
Decision Management Community Challenge Feb 2018- Tax Decision Table
A solution using Camunda

(Bob Moore, JETset Business Consulting, 25 Feb 2018)

1 Problem Statement (cut and pasted from the web site)

With tax season upon us, take a look at the following decision table created by a business analyst from a tax agency:



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	**IF This year's filing**	1	2	3	4	5	6	7	8	9	10	11	12	
2	Doc Code being processed is 7,8 or 28	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
3	Taxable Income < 50,000	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
4	Interest Income < 400	-	N	N	N	N	-	-	-	-	Y	Y	Y	
5	Taxable Pension = 0	-	-	-	-	-	N	N	Y	Y	Y	Y	Y	
6	Under 65 Years Old	-	Y	N	N	Y	N	N	N	N	Y	Y	Y	
7	AGI < 25,000	-	-	N	Y	-	N	Y	N	Y	-	-	-	
8	Doc Code being processed is a 29	-	-	-	-	-	-	-	-	-	Y	N	N	
9	Zip Code OK (not SC, APO or LL)	-	-	-	-	-	-	-	-	-	-	Y	-	
10	**THEN**													
11	Next Year send the taxpayer the following forms and schedules: 1040EZ Telefile (MFR 15)										Y	Y		
12	Next Year send the taxpayer the following forms and schedules: 1040EZ (MFR 13)												Y	
13	Next Year send the taxpayer the following forms and schedules: 1040A, with Sch 1,2 (MFR 02)	Y	Y	Y		Y	Y		Y					
14	Next Year send the taxpayer the following forms and schedules: 1040A with Sch R (MFR 14)				Y			Y		Y				

How would you represent this decision table using your favorite business rules and decision management tool? Submit your decision table(s) to us and our readers will be able to compare different representations (and the tools).

2 Meeting the Challenge

I was in two minds about having a go at this, it seemed a bit trivial as an exercise, but having found Camunda[1] an accessible and easy to use tool for drawing DRDs, I thought I might try out how well it copes with the other of James Taylor's "Core DMN Components" – decision tables[2].

It turned out to very simple in doing to do this. I had a few minor challenges in terms of getting the decision table to execute but starting from scratch and with no knowledge of how to use Camunda either for building decision tables, or for deploying one to their execution engine, I had things up and running in a few hours. The resulting table is shown below.

---

[1] See https://camunda.org/

[2] See https://dmcommunity.org/2018/01/16/core-dmn-components/

Note that in creating the decision table, I've transformed it from a vertical to horizontal orientation and replaced 'Y' values with 'proper' conditions and output values.

FormsSchedules

Enter Advanced Mode | Show DRD

U	Input +								Output +	Annotation
	DocumentCode	TaxableIncome	InterestIncome	TaxablePension	Age	AGI	ZipCode	DocumentCode2	FormsSchedules	
	string	integer	integer	integer	integer	integer	string	string	string	
1	"7", "8", "28"	>= 50000	-	-	-	-	-	-	"1040A wth Sch 1,2 (MFR 02)"	Column B
2	"7", "8", "28"	< 50000	>=400	-	< 55	-	-	-	"1040A wth Sch 1,2 (MFR 02)"	Column C
3	"7", "8", "28"	< 50000	>= 400	-	>= 55	>= 25000	-	-	"1040A with Sch R (MFR 14)"	Column D
4	"7", "8", "28"	< 50000	>=400	-	>= 55	< 25000	-	-	"1040A wth Sch 1,2 (MFR 02)"	Column E
5	"7", "8", "28"	< 50000	>= 400	-	< 55	-	-	-	"1040A wth Sch 1,2 (MFR 02)"	Column F
6	"7", "8", "28"	< 50000	-	> 0	>= 55	>= 25000	-	-	"1040A with Sch R (MFR 14)"	Column G
7	"7", "8", "28"	< 50000	-	> 0	>= 55	< 25000	-	-	"1040A wth Sch 1,2 (MFR 02)"	Column H
8	"7", "8", "28"	< 50000	-	0	>= 55	>= 25000	-	-	"1040A with Sch R (MFR 14)"	Column I
9	"7", "8", "28"	< 50000	-	0	>= 55	< 25000	-	-	"1040EZ Telefile (MFR 15)"	Column J
10	"7", "8", "28"	< 50000	< 400	0	< 55	-	-	"28"	"1040EZ Telefile (MFR 15)"	Column K
11	"7", "8", "28"	< 50000	< 400	0	<55	-	not("SC", "APO", "LL")	not("28")	"1040A wth Sch 1,2 (MFR 02)"	Column L
12	"7", "8", "28"	< 50000	< 400	0	< 55	-	-	not("28")	"1040EZ (MFR 13)"	Column M

Decision Table Based on Straightforward translation of Spreadsheet (including some problems!)

3 Discussion

As described the prime objective of the challenge was to see how various tools implement the table. As regards using Camunda, it proved very simple and straightforward. For 'simple' decision tables like this the tool is intuitive to use, and it takes little time to get things done.

There do seem to be some limitations with regard to the DMN standard. In particular, it seems that only 'horizontal orientation' is supported. This is not for me a big issue as I prefer this to vertical or mixed orientations (the spreadsheet of course uses vertical orientation). I do have a bit of an issue with runtime semantics as I describe below.

Overall, I was as pleased with Camunda on my second exploration of its capabilities as I was on my first. It's not the most complete or sophisticated tool available, but it did all I wanted it to. Unfortunately, (as presented above) my solution doesn't quite work.

To see why one can look at the only other solution to the challenge (at the time of writing) – Mike Parrish's implementation in Corticon[3]. As we know, everyone addressing a challenge, takes somewhat different perspective of what the "solution" should look like. I blandly implemented what I saw on the spreadsheet, but Mike looked for and found several flaws in the logic expressed by the spreadsheet.

Several issues are obvious from only a cursory examination. There is incompleteness. In row 2 all the cells are filled with 'Y'. The spreadsheet tells us what to do with document codes 7, 8 and 28, but what if the document code is not one of these? There is redundancy. The contents of column C and column F are identical, so evidently one of them is unneeded. Last, and perhaps worst, there is inconsistency. Any inputs satisfying rule 11 also satisfy rule 12 which gives a different outcome.

The ease with which Mike was able to identify these and other issues emphases that tools like Corticon offer significantly more functionality than Camunda in this area. At runtime Camunda does detect some inconsistencies, but not how I'd like it to. Running my decision table, I quickly found that if you specify a table is unique, then an exception is thrown if your input data causes more than one rule to fire. From an architectural perspective I can't say I like the idea of build time specification errors in a decision service giving rise to runtime exceptions. And clearly it is much better to detect, report and fix such inconsistencies at build time rather than at runtime, particularly if they only arise for rare kinds of input.

I agree with all the shortcomings Mike identified in his analysis of the spreadsheet and have to admit on the whole I'd probably rather I'd been able to Coritcon than Camunda to address the challenge. However I wasn't, and so I found myself wanting answers to two questions. Firstly, how could I 'fix' my version of the decision table (ideally without trying to copy what Mike had done)? Secondly, how did the analyst get away with doing such a bad job? After a bit of thought, I came to understand that to answer the first question, perhaps I should assume that the answer to the second was that the analyst did ***not*** do a bad job after all. Suppose the apparent inconsistencies and incompleteness are not because the analyst's rules are 'wrong', but because we are misinterpreting the spreadsheet?

---

[3] See https://dmcommunity.files.wordpress.com/2018/02/challengetaxtablemikeparish.pdf

4 A 'Corrected' Solution

By default, DMN - and (it seems) many in the decision management community - assume decision tables use a 'unique' hit policy. My experience is that most people who are not decision management practitioners assume a 'first' hit policy. They look down a table until they find a row matching their input data and just take that result. The fact that another row further down the table might also match (possibly with a different result) is of no concern to them. If the analyst was assuming first hit semantics (which seems the likely case) inconsistencies in the rules just don't occur[4]. For example, rule 12 no longer subsumes rule 11, because first hit semantics give it an implicit condition that the zip code *is* one of SC, APO or LL, because it is after rule 11 in the table.

Now what about incompleteness? Disregarding the issue about document types other than 7, 8 and 28, incompleteness seems to narrow down to rules from 6 to 12. There are no rules about people under the age threshold who have a 'taxable pension'. My guess is that this is because the analyst has reason to believe such people do not exist.

When I looked at the fuzzy image of the spreadsheet at first I instinctively assumed cell A6 was talking about being under 65 years of age, but looking again I decided it was a bit unclear and zooming in decided on balance the age threshold is 55 not 65 (Mike's solution indicates he read this as 65, which I can't argue with given the quality of the image). Applying a little domain knowledge, if you are over 65 you may well have a pension income relevant for tax purposes, and it's plausible you might have one if you are over 55. But if you are under 55, whatever income you have, my guess is the IRS is unlikely to accept it is a (retirement) pension (the UK tax laws certainly don't).

'Taxable pension' is a vague term, and we can't be sure what it means. But assuming the age threshold is 55, and also that the spreadsheet is broadly 'correct', it strongly suggests that 'taxable pension' is only of interest if you are 55 or above. This would mean the zeros in rows 10 to 12 under "TaxablePension" are redundant. The analyst's logic is that if you are under 55 your 'taxable pension' is going to be zero anyway. This means we can either ignore the incompleteness, since we 'know' we should never find anyone under 55 with a "taxable pension", or simply remove the "TaxablePension" values for these rows. Either way we can now resolve the incompleteness issues.

What of redundancy? When programming, redundancy is generally bad, but, if acknowledged, it can be useful in specifications. Franklin's observation that taxation is an inevitable as death now has a corollary. Annual changes to tax law seem just as inevitable. I suspect the redundancies are memories of past and/or portents of future tax rules. Perhaps the analyst felt they would be useful to leave in to be ready for next year's changes? I wouldn't encourage this behaviour, but is it really so bad?

So, is the spreadsheet fundamentally flawed or not? We just don't know. With just an image of it, and no explanatory text or other contextual information it's impossible to tell[5]. However, I'd like to believe the analyst knew what he (or she) was doing. We are missing the knowledge needed to fully interpret it, but assuming the analyst did know what they were doing we can make a fair guess at what it was intended to convey.

---

[4] I don't advocate 'first hit' over 'unique', I'd prefer to be able to check consistency! I'm just trying to reflect the way the analyst most likely captured the tax rules (and how 'most' people would interpret them).

[5] Especially since we can't even read it well enough to agree on what it says!

Based on what I feel are the reasonable assumptions above, I 'corrected' my version of the decision table (see below), simply by making it 'first hit', and removing the dependence on Taxable Pension from rules 10, 11 & 12[6]. This gets rid of the inconsistencies and incompleteness (if not the redundancies) and I think there is a good chance that this version correctly captures the intent of the business analyst, while remaining very close to the original spreadsheet. I'm not going to argue this is the best way to express things, but it is logically sound.

However, as Mike justly emphasises it would be essential to go back and check. Just because assumptions are reasonable doesn't make them right!

FormsSchedules1

FormsSchedules1

Exit Advanced Mode | Show DRD

F	Input +								Output +	Annotation
	DocumentCode	TaxableIncome	InterestIncome	TaxablePension	Age	AGI	ZipCode	DocumentCode2	FormsSchedules	
	documentCode	taxableIncome	interestIncome	taxablePension	age	agi	zipCode	documentCode	formsSchedules	
	string	integer	integer	integer	integer	integer	string	string	string	
1	"7", "8", "28"	>= 50000	-	-	-	-	-	-	"1040A wth Sch 1,2 (MFR 02)"	Column B
2	"7", "8", "28"	< 50000	>=400	-	< 55	-	-	-	"1040A wth Sch 1,2 (MFR 02)"	Column C
3	"7", "8", "28"	< 50000	>= 400	-	>= 55	>= 25000	-	-	"1040A with Sch R (MFR 14)"	Column D
4	"7", "8", "28"	< 50000	>=400	-	>= 55	< 25000	-	-	"1040A wth Sch 1,2 (MFR 02)"	Column E
5	"7", "8", "28"	< 50000	>= 400	-	< 55	-	-	-	"1040A wth Sch 1,2 (MFR 02)"	Column F
6	"7", "8", "28"	< 50000	-	> 0	>= 55	>= 25000	-	-	"1040A with Sch R (MFR 14)"	Column G
7	"7", "8", "28"	< 50000	-	> 0	>= 55	< 25000	-	-	"1040A wth Sch 1,2 (MFR 02)"	Column H
8	"7", "8", "28"	< 50000	-	0	>= 55	>= 25000	-	-	"1040A with Sch R (MFR 14)"	Column I
9	"7", "8", "28"	< 50000	-	0	>= 55	< 25000	-	-	"1040EZ Telefile (MFR 15)"	Column J
10	"7", "8", "28"	< 50000	< 400	-	< 55	-	-	"28"	"1040EZ Telefile (MFR 15)"	Column K
11	"7", "8", "28"	< 50000	< 400	-	<55	-	not("SC", "APO", "LL")	not("28")	"1040A wth Sch 1,2 (MFR 02)"	Column L
12	"7", "8", "28"	< 50000	< 400	-	< 55	-	-	not("28")	"1040EZ (MFR 13)"	Column M

"Corrected" Decision Table

[6] On the basis that if you are under 55 your 'taxable pension' will be zero, I could simply have changed from 'unique' to 'first' in the original decision table, giving me a solution which is essentially identical to the original but for the change of a single letter in the left hand corner, but I felt at least a little effort to tidy up was obligatory.