Unlocking the Full Potential of DMN

DecisionCAMP2021

Simon Vandevelde, Joost Vennekens
KU Leuven - EAVISE
15 September 2021

AI FLANDERS
BUILDING OUR DIGITAL FUTURE

KU LEUVEN

What this presentation is

Unlocking the Full Potential of DMN

... a bit click-baity

What this presentation is

Different ways to apply the knowledge in DMN models

Goal of the presentation

Goal

Show that DMN can be used in many more ways

→ by deviating from the “standard” approach used with DMN

Outline

1. DMN: now
2. Execution methods
3. Alternative Execution Methods
4. Practical systems
5. Conclusion

1 DMN: NOW

DMN: NOW

Lifetime of DMN models:

- Creation of the model
- Execution of the model

Much research interest goes to **creation!**

DMN: CREATION

Creation topics:

- “Knowledge Acquisition”
  - Domain expert
  - Decision mining
  - . . .
- Modeling methodologies
- Verification & Validation
  - Automated testing
- . . .

DMN: EXECUTION

Execution topics:

- Interaction with DMN
  - Chat bots
  - Microservices
  - “Traceability”
  - . . .
- Efficiency of execution
- **Execution methods**

2 Execution methods

Current Execution Methods

Some of the current execution methods:

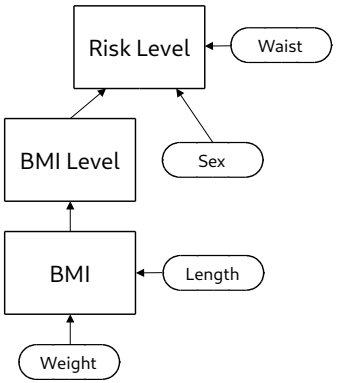
- The standard: *bottom-to-top execution*
- Reasoning on sub-decisions
- “Wildcard mode”

Bottom-to-top

"The standard approach", works in three steps:

1. Start at bottom inputs
2. Evaluate tables one by one
3. Finish at top decisions

Bottom-to-top: example



Risk Level

U	BMILevel	Sex	Waist	RiskLevel
1	Normal	–	–	Low
2	Underweight	–	–	High
3	Overweight	Male	≤ 102	Increased
4	Overweight	Male	> 102	High
5	Overweight	Female	≤ 88	Increased
6	Overweight	Female	> 88	High
7	Obese	Male	≤ 102	High
8	Obese	Male	> 102	Very High
9	Obese	Female	≤ 88	High
10	Obese	Female	> 88	Very High

BMI Level

U	BMI	BMILevel
1	< 18.5	Underweight
2	$[18.5..25]$	Normal
3	$(25..30]$	Overweight
4	> 30	Obese

BMI

U	Weight	Length	BMI
1	–	–	Weight/(Length*Length)

Bottom-to-top: example

Four inputs:

- Weight
- Length
- Sex
- Waist

→ RiskLevel

Risk Level

U	BMILevel	Sex	Waist	RiskLevel
1	Normal	–	–	Low
2	Underweight	–	–	High
3	Overweight	Male	≤ 102	Increased
4	Overweight	Male	> 102	High
5	Overweight	Female	≤ 88	Increased
6	Overweight	Female	> 88	High
7	Obese	Male	≤ 102	High
8	Obese	Male	> 102	Very High
9	Obese	Female	≤ 88	High
10	Obese	Female	> 88	Very High

BMI Level

U	BMI	BMILevel
1	< 18.5	Underweight
2	$[18.5..25]$	Normal
3	$(25..30]$	Overweight
4	> 30	Obese

BMI

U	Weight	Length	BMI
1	–	–	Weight/(Length*Length)

Bottom-to-top: example

Male, 90kg, 1.79m,
Waist = 100cm:

1. *BMI*
   $\rightarrow$ *28*
2. *BMILevel*
   $\rightarrow$ *Overweight*
3. *RiskLevel*
   $\rightarrow$ *Increased*

Risk Level

U	BMILevel	Sex	Waist	RiskLevel
1	Normal	–	–	Low
2	Underweight	–	–	High
3	Overweight	Male	≤ 102	Increased
4	Overweight	Male	> 102	High
5	Overweight	Female	≤ 88	Increased
6	Overweight	Female	> 88	High
7	Obese	Male	≤ 102	High
8	Obese	Male	> 102	Very High
9	Obese	Female	≤ 88	High
10	Obese	Female	> 88	Very High

BMI Level

U	BMI	BMILevel
1	< 18.5	Underweight
2	$[18.5..25]$	Normal
3	$(25..30]$	Overweight
4	> 30	Obese

BMI

U	Weight	Length	BMI
1	–	–	Weight/(Length*Length)

Sub-Decision

- Only interested in a subsection of the model
- However, for bottom-to-top, all inputs are needed
- Solution: reasoning on sub-decision

E.g., evaluating a single table at a time

Sub-Decision: Example

Inputs for BMILevel

- Weight
- Length

OR

- BMI

Risk Level

U	BMILevel	Sex	Waist	RiskLevel
1	Normal	–	–	Low
2	Underweight	–	–	High
3	Overweight	Male	≤ 102	Increased
4	Overweight	Male	> 102	High
5	Overweight	Female	≤ 88	Increased
6	Overweight	Female	> 88	High
7	Obese	Male	≤ 102	High
8	Obese	Male	> 102	Very High
9	Obese	Female	≤ 88	High
10	Obese	Female	> 88	Very High

BMI Level

U	BMI	BMILevel
1	< 18.5	Underweight
2	$[18.5..25]$	Normal
3	$(25..30]$	Overweight
4	> 30	Obese

BMI

U	Weight	Length	BMI
1	–	–	Weight/(Length*Length)

Wildcard mode

- Reason on partial information
- Set unknown inputs as “wildcard”
- System outputs all possible outcomes
- Supported by Signavio and Camunda

Wildcard mode: Example

90kg, 1.79m,
Waist = 100cm, *Sex* unknown

- RiskLevel
  - $\rightarrow$ Increased
  - $\rightarrow$ High

Risk Level

U	BMILevel	Sex	Waist	RiskLevel
1	Normal	–	–	Low
2	Underweight	–	–	High
3	Overweight	Male	≤ 102	Increased
4	Overweight	Male	> 102	High
5	Overweight	Female	≤ 88	Increased
6	Overweight	Female	> 88	High
7	Obese	Male	≤ 102	High
8	Obese	Male	> 102	Very High
9	Obese	Female	≤ 88	High
10	Obese	Female	> 88	Very High

BMI Level

U	BMI	BMILevel
1	< 18.5	Underweight
2	$[18.5..25]$	Normal
3	$(25..30]$	Overweight
4	> 30	Obese

BMI

U	Weight	Length	BMI
1	–	–	Weight/(Length*Length)

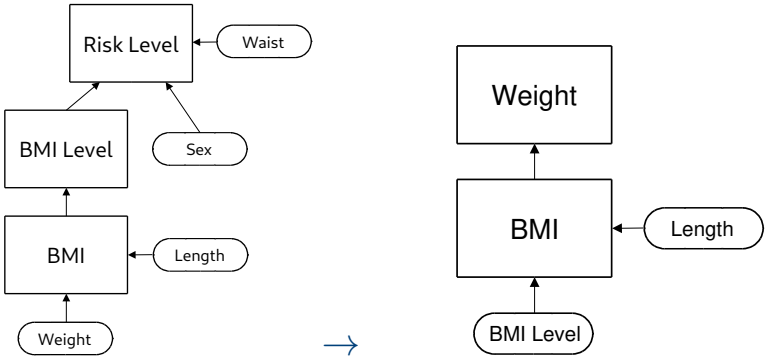
3 Alternative Execution Methods

Multidirectional Reasoning: example

- We have calculated *BMILevel = Overweight*
- Logical next question:
  - "What should weight be for healthy BMI?"
- Easy solution: *guess* until you find correct answer
  - Only easy for small models
  - Never 100% precise

→ Apply DMN model "in reverse"!

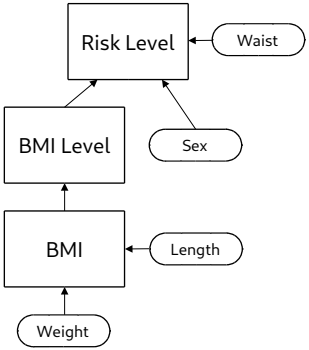
Multidirectional Reasoning: example



note: only shown as example, this is not actually correct.

Multidirectional Reasoning: DRD

- Actually, no need to change DRD
- DRD does not show execution flow!
- Rather, the order of definitions



Multidirectional Reasoning

Re-using DMN in different directions:

- Endless new uses for model
- Re-use existing knowledge
- Easy answer for hypothetical “but-now-what” questions

$\rightarrow$ But how do we do it practically?

Multidirectional Reasoning in practice

In theory: *simply* reverse DMN tables
In practice:

BMI Level		
U	BMI	BMILevel
1	< 18.5	Underweight
2	$[18.5..25]$	Normal
3	$(25..30]$	Overweight
4	> 30	Obese

BMI			
U	Weight	Length	BMI
1	–	–	Weight/(Length*Length)

BMI Level		
U	BMILevel	BMI
1	Underweight	< 18.5
2	Normal	$[18.5..25]$
3	Overweight	$(25..30]$
4	Obese	> 30

BMI			
U	BMI	Length	Weight
1	–	–	BMI*Length*Length

note: this is not actually correct, as it changes the semantics of the model

Multidirectional Reasoning in practice

In theory: *simply* reverse DMN tables
In practice:

Output can now be:

- Range
- List of elements
- “—”?

BMI Level		
U	BMILevel	BMI
1	Underweight	< 18.5
2	Normal	$[18.5..25]$
3	Overweight	$(25..30]$
4	Obese	> 30

BMI			
U	BMI	Length	Weight
1	–	–	BMI*Length*Length

note: this is not actually correct, as it changes the semantics of the model

Multidirectional Reasoning in practice

In theory: *simply* reverse DMN tables
In practice:

Output can now be:
$\rightarrow$ constraints!
$\rightarrow$ not DMN compliant!

BMI Level		
U	BMILevel	BMI
1	Underweight	< 18.5
2	Normal	$[18.5..25]$
3	Overweight	$(25..30]$
4	Obese	> 30

BMI			
U	BMI	Length	Weight
1	–	–	BMI*Length*Length

note: this is not actually correct, as it changes the semantics of the model

Multidirectional Reasoning in practice

In the case of range or list:

- Multiple solutions possible
- Any solution is equally correct
- Value of variables can be enumerations

Multidirectional Reasoning in practice

1.79m & Normal *BMILevel*:
$\rightarrow$ *BMI* $= [18.5..25]$
$\rightarrow$ *Weight* $= [59..80]$kg

BMI Level

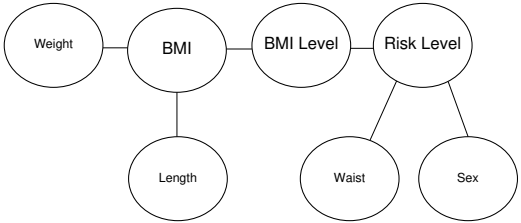
U	BMILevel	BMI
1	Underweight	< 18.5
2	Normal	$[18.5..25]$
3	Overweight	$(25..30]$
4	Obese	> 30

BMI

U	BMI	Length	Weight
1	–	–	BMI*Length*Length

Multidirectional Reasoning

We can do more than “reasoning in reverse”!
$\rightarrow$ we can reason from any variables to any other



... just not always meaningful

Multidirectional Reasoning: other examples

E.g. consider DMN to calculate pension based on age, worked years, ...
$\rightarrow$ “When can I retire if I want an income of x per month?”

A ticket price calculator for traveling to place x:
$\rightarrow$“At what time will the ticket be cheapest?”

Multidirectional Reasoning: other examples

In a system to check eligibility for bonuses:
→“What can I do to gain more bonuses?”

In the case of loans & interests:
→“What should I do to have interest rate of 1%?”

Optimization

In BMI example, we asked:

What should my weight be in order to be healthy?

But actually, we ask the following:

What is the highest weight I can have in order to be healthy?

$\rightarrow$ Optimization!

Optimization

1.79m & Normal *BMILevel*:

$\rightarrow$ *BMI* $= [18.5..25]$

$\rightarrow$ *Weight* $= [59..80]$kg

Max: 80kg

Min: 59kg

BMI Level		
U	BMILevel	BMI
1	Underweight	< 18.5
2	Normal	$[18.5..25]$
3	Overweight	$(25..30]$
4	Obese	> 30

BMI			
U	BMI	Length	Weight
1	–	–	BMI*Length*Length

Relevance

Sometimes, variables might not be relevant

E.g. if *BMILevel = Normal*
$\rightarrow$ *Sex* and *Waist* irrelevant

Risk Level				
U	BMILevel	Sex	Waist	RiskLevel
1	Normal	–	–	Low
2	Underweight	–	–	High
3	Overweight	Male	≤ 102	Increased
4	Overweight	Male	> 102	High
5	Overweight	Female	≤ 88	Increased
6	Overweight	Female	> 88	High
7	Obese	Male	≤ 102	High
8	Obese	Male	> 102	Very High
9	Obese	Female	≤ 88	High
10	Obese	Female	> 88	Very High

Relevance

Sometimes, variables might not be relevant

I.e., if $BMI \leq 25$
$\rightarrow$ no need for *Sex* or *Waist*

Risk Level				
U	BMILevel	Sex	Waist	RiskLevel
1	Normal	–	–	Low
2	Underweight	–	–	High
3	Overweight	Male	≤ 102	Increased
4	Overweight	Male	> 102	High
5	Overweight	Female	≤ 88	Increased
6	Overweight	Female	> 88	High
7	Obese	Male	≤ 102	High
8	Obese	Male	> 102	Very High
9	Obese	Female	≤ 88	High
10	Obese	Female	> 88	Very High

4 Practical systems

Implementation

How to implement these novel execution methods?
→ we can't use current execution engines
→ "reversing" a model is not DMN-compliant!
→ How to best deal with constraints?

Solution:

Use a constraint solver!

Implementation examples

Two applications supporting our execution approach:

- **DMN-IDP**: combines DMN with Knowledge-Based AI in interface
- **DMN-IDPy**: similar, but in form of Python API

$\rightarrow$ Both use the KU Leuven's IDP system[1] as constraint solver

[1] https://dtai.cs.kuleuven.be/software/idp

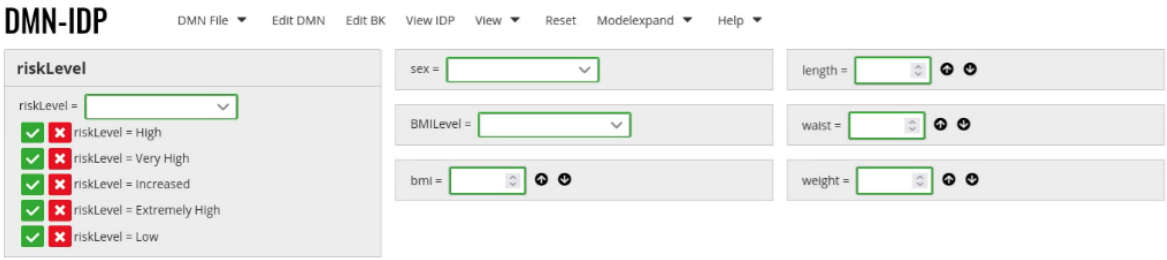
DMN-IDP[1]

Combines DMN editor with our IDP-based interface[2]

- DMN models are translated into IDP knowledge base
- Interface supports:
  - bottom-to-top
  - multidirectional reasoning
  - optimization
  - reasoning on incomplete information

[1] Vandevelde, Simon, and Joost Vennekens. 2020. 'A Multifunctional, Interactive DMN Decision Modelling Tool'.
[2] Carbonnelle, Pierre, Bram Aerts, Marjolein Deryck, Joost Vennekens, and Marc Denecker. 2019. 'An Interactive Consultant'.

DMN-IDP: DEMO


DMN-IDP
DMN File
Edit DMN
Edit BK
View IDP
View
Reset
Modelexpand
Help
riskLevel
riskLevel =
riskLevel = High
riskLevel = Very High
riskLevel = Increased
riskLevel = Extremely High
riskLevel = Low
sex =
BMILevel =
bmi =
length =
waist =
weight =

DMN-IDPy[1]

Similar to DMN-IDP, but in API form

- Load DMN xml in Python
- Features:
  - bottom-to-top
  - reasoning on incomplete information
  - relevance
  - multidirectional reasoning
  - querying if a variable is “known”
  - querying variable type and possible values
  - optimization

[1] Vandevelde, Simon, Vedavyas, Etikala, Vanthienen, Jan, Vennekens, Joost. 2021 “Leveraging the Power of IDP with the Flexibility of DMN: a Multifunctional API”. Proceedings of RuleML+RR21.

DMN-IDPy: DEMO

```
# Import the API (This might take some time)
from cdmn.API import DMN
spec = DMN(path='./DMN/BMILevel_paper.dmn', auto_propagate=True)

variables = spec.get_outputs() + spec.get_intermediary()  # We won't be calculating inputs
requested_variable = input('Which variable would you like to calculate? {}\n>'.format(variables))

# Generate the list of missing input variables needed to calculate the requested variable.
missing_variables = [x for x in spec.dependencies_of(requested_variable)
                     if x in spec.get_inputs()
                     and x in spec.get_unknown_variables()]
print("\nThe following variables are still unknown:")
print(missing_variables)
for var in missing_variables:
    # Ask for the variable's value. Based on var type, ask different question.
    var_type = spec.type_of(var)
    if var_type in ['Real', 'Int']:
        msg = "Value for {} ({}) unknown.\nSet value as:\n>".format(var, spec.type_of(var))
    else:
        msg = "Value for {} unknown.\nPossible values: [{}]\n>".format(var, spec.possible_values_of(var))
    value = input(msg)
    spec.set_value(var, value)

print('Calculated value for {}:\n{}'.format(requested_variable, spec.value_of(requested_variable)))
```

DMN-based AI?

Can DMN be used as knowledge base for symbolic AI?

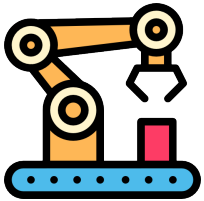
- user-friendly
- readable
- sufficiently expressive

$\rightarrow$ but current execution methods are limited

DMN-based AI? Example

Imagine a cobot

- Assembly line for product x
- Cobot and operator work together
- Cobot has a preset procedures

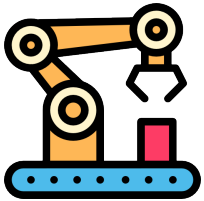


Icons made by Freepik from www.flaticon.com

DMN-based AI? Example

Imagine a cobot

- "Screw in y"
- "Verify weld line quality"
- "Give operator screwdriver if z happens"

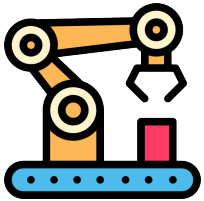


Icons made by Freepik from www.flaticon.com

DMN-based AI? Example

Imagine a cobot

- This can likely be expressed in one model
- But does standard functionality suffice?

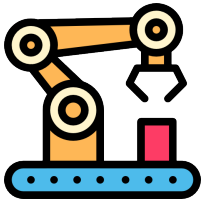


Icons made by Freepik from www.flaticon.com

DMN-based AI? Example

Imagine a cobot

- Perform task-by-task: sub-decision
- “Find all situations x in which the robot will turn past the operator”: multidirectional
- Only turn on necessary sensors: relevancy

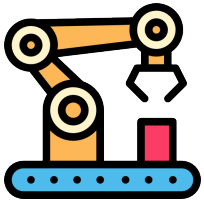


Icons made by Freepik from www.flaticon.com

DMN-based AI? Example

Imagine a cobot

- How to prepare for procedure y?: multidirectional
- Use optimal number of screws: optimize
- . . .



Icons made by Freepik from www.flaticon.com

5 Conclusion

Conclusion

- We can do more with DMN, without changing it!
- By adding support for more flexible execution
  - reasoning on incomplete information
  - multidirectional reasoning
  - optimization
  - “wildcard”
- A single model can have multiple purposes!
- Huge potential in combination with AI

Thank you for your attention.

Any questions?

For further questions or discussion:
s.vandevelde@kuleuven.be