* DMN: how to satisfy multiple objectives?

Prof. dr. Jan Vanthienen, KU Leuven, FEB

* Who?

Jan Vanthienen
KU Leuven
Faculty of Economics and Business
Business Information Systems Group

Research and teaching:
- Business rules, processes and information systems
- Information & Knowledge Management
- Decision models & tables
- Business Intelligence & Analytics

IBM Faculty Award
Belgian Francqui Chair 2009 at FUNDP
RuleML best paper award 2007

- Bpost bank Research Chair Actionable Analytics
- Colruyt-Symeta Research Chair Smart Data and Decisions in Marketing
- IBM Fund Intelligent Business Decision Making
- Microsoft Research Chair on Intelligent Environments
- PricewaterhouseCoopers Chair on E-Business

Email: jan.vanthienen@kuleuven.be
**Abstract**

* Decision modeling with DMN allows to model decisions and offers a standard notation and expression for decision requirements and decision logic.

* Is there only one best model for a (set of) decision(s) or can you have multiple models?

* Based on a recent Decision Management Community challenge, this presentation investigates various objectives of decision modeling:
  
  • providing an overview for business,
  • verification of business logic,
  • traceability to knowledge sources,
  • maintainability,
  • model-driven execution.

---

**Problem statement**

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.

Notes

• 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).
There was a large variety of tools and formats. It would be good for exchange and readability if the common notation is used:

* [45..60) or [45..60] are well-defined notations (at least one of them) 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 was not always clear from the submissions what was 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.
**Typical solution 1 (Multiple Hit)**

### What?
- Collect/add all rules that apply

### Pro
- Traceability +/-

### Con
- Avoiding 5+2 is work
- V&V: Who gets 5+3?
- Where is 'my case'?

---

**Decision Table Multiple Hit Define Vacation Days**

<table>
<thead>
<tr>
<th>Condition</th>
<th>Condition</th>
<th>Vacation Days</th>
</tr>
</thead>
<tbody>
<tr>
<td>Age in Years</td>
<td>Years of Service</td>
<td>= 22</td>
</tr>
<tr>
<td>&lt; 18</td>
<td>&gt; = 30</td>
<td>+ = 5</td>
</tr>
<tr>
<td>&gt;= 18</td>
<td>&gt; = 30</td>
<td>+ = 5</td>
</tr>
<tr>
<td>&gt; = 60</td>
<td>&lt; 30</td>
<td>+ = 3</td>
</tr>
<tr>
<td>&gt; = 60</td>
<td>&gt; = 30</td>
<td>+ = 3</td>
</tr>
<tr>
<td>Within [45, 60]</td>
<td>&lt; 30</td>
<td>+ = 2</td>
</tr>
<tr>
<td>&lt; 45</td>
<td>Within [15, 30]</td>
<td>+ = 2</td>
</tr>
</tbody>
</table>

---

**Rule Statements**

<table>
<thead>
<tr>
<th>Ref</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Base: Every employee receives at least 22 days. Extra days:</td>
</tr>
<tr>
<td>1</td>
<td>1A. Only employees younger than 18 receive 5 extra days.</td>
</tr>
<tr>
<td>2</td>
<td>2B. Only employees at least 60 receive 5 extra days.</td>
</tr>
<tr>
<td>3</td>
<td>3C. Only employees with at least 30 years of service receive 5 extra days (if they didn't already get 5 days because of their age).</td>
</tr>
<tr>
<td>4</td>
<td>4D. Employees with at least 30 years of service and also employees of age 60 or more receive 2 extra days on top of other days.</td>
</tr>
<tr>
<td>5</td>
<td>5A. If an employee has at least 15 but less than 30 years of service, 2 extra days are given. They cannot be combined with the 5 extra days.</td>
</tr>
<tr>
<td>6</td>
<td>6B. These 2 days are also provided for employees age 45 or more. They cannot be combined with the 5 extra days.</td>
</tr>
<tr>
<td>7</td>
<td>Employees do not get extra days when years of service is less than 15 and age is 18-44.</td>
</tr>
</tbody>
</table>
**Typical solution 2 (Any)**

What?
- Take any rule that applies (result =)

Pro
- Traceability +/-

Con
- Avoiding \(5+2\) is work
- Completeness?
- Overlap/Redundancy

**Typical solution 3 (First)**

What?
- Hit & run

Pro
- Traceability +/-

Con
- Who gets \(22+2\)?
- Hard to maintain
- Completeness?

---

(c) Jan Vanthienen

Decision Camp 2016
What?

* Take the one rule that applies

<table>
<thead>
<tr>
<th>Age in Years</th>
<th>Years of Service</th>
<th>Vacation Days</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;18</td>
<td></td>
<td>(22 + 6)</td>
</tr>
<tr>
<td>(18.45)</td>
<td>&lt;15</td>
<td>(22 + 2)</td>
</tr>
<tr>
<td>(18.45)</td>
<td>&gt;=30</td>
<td>(22 + 5 + 3)</td>
</tr>
<tr>
<td>(45.80)</td>
<td>&lt;15</td>
<td>(22 + 2)</td>
</tr>
<tr>
<td>(45.80)</td>
<td>(15.30)</td>
<td>(22 + 2)</td>
</tr>
<tr>
<td>(45.80)</td>
<td>&gt;=30</td>
<td>(22 + 5 + 3)</td>
</tr>
<tr>
<td>60+</td>
<td></td>
<td>(22 + 5 + 3)</td>
</tr>
</tbody>
</table>

Initial Action
Days_leave = MV 22

V&V
* Completeness
Con
* Traceability +/-
**Solutions with Decomposition**

**Base Days Decision**

<table>
<thead>
<tr>
<th>Base Days</th>
<th>Eligible for Extra 5 Days</th>
<th>Eligible for Extra 3 Days</th>
<th>Eligible for Extra 2 Days</th>
<th>Vacation Days</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td>22</td>
</tr>
<tr>
<td>3</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>False</td>
<td>False</td>
<td>False</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td>21</td>
</tr>
</tbody>
</table>

**Extra 2 Days Decision**

<table>
<thead>
<tr>
<th>Extra 2 Days</th>
<th>Age</th>
<th>Years of Service</th>
<th>Eligible</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Age</td>
<td>Years of Service</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>=55</td>
<td>&gt;30</td>
<td>False, true</td>
</tr>
<tr>
<td>2</td>
<td>=55</td>
<td>&gt;30</td>
<td>True</td>
</tr>
</tbody>
</table>

**Pro**

- Traceability
- Simple

**Con**

- Overview

---

**Solutions with Decomposition**

**Total Vacation Days**

<table>
<thead>
<tr>
<th>Total Vacation Days</th>
<th>Base Days</th>
<th>Eligible for Extra 5 Days</th>
<th>Eligible for Extra 3 Days</th>
<th>Eligible for Extra 2 Days</th>
<th>Vacation Days</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td>False</td>
<td>22</td>
</tr>
<tr>
<td>3</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>False</td>
<td>False</td>
<td>False</td>
<td>False</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>21</td>
</tr>
</tbody>
</table>

**Decision Camp 2016**

(c) Jan Vanthienen
**Some DMN features**

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 the 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’).

---

**Separating eligibility - exclusion**

---
Model or execution?

Default is 30
Compact (!) solution (?)

<table>
<thead>
<tr>
<th>Employee.Age &gt;= 60 OR Employee.YearsOfService &gt;= 30</th>
<th>TRUE</th>
<th>FALSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Employee.Age &lt; 18</td>
<td>*</td>
<td>TRUE</td>
</tr>
<tr>
<td>(Employee.YearsOfService &gt;= 15 AND Employee.YearsOfService &lt; 30) OR Employee.Age &gt;= 45</td>
<td>*</td>
<td>TRUE</td>
</tr>
<tr>
<td>Employee.VacationDays</td>
<td>30</td>
<td>27</td>
</tr>
</tbody>
</table>

Business questions and criteria

* 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? Where is my case?

* Analysis: Are there some strange assignments in giving additional days? Yes, if you turn 18, e.g., you lose 5 bonus days.

* Maintenance: What if the combination policy for 2 and 5 bonus days changes?

* Traceability: How easy is it to trace back to the original text (rules)?
Solution Types

* 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.

* Building 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.

* 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.

Modeling objectives

Specification
(description, modelling)

Verification & Validation

Execution
(implementation)
### Solutions

<table>
<thead>
<tr>
<th>Table type?</th>
<th>DRD?</th>
<th>Traceable</th>
<th>Maintainable</th>
<th>Overview</th>
<th>DMN conformant</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unique</td>
<td>medium high</td>
<td>medium high</td>
<td>medium high</td>
<td>high</td>
<td>very good</td>
<td></td>
</tr>
<tr>
<td>Multiple</td>
<td>medium high</td>
<td>medium high</td>
<td>medium high</td>
<td>high</td>
<td>good</td>
<td></td>
</tr>
<tr>
<td>First Decomposed</td>
<td>high medium</td>
<td>medium medium</td>
<td>high medium</td>
<td>medium high</td>
<td>excellent</td>
<td></td>
</tr>
<tr>
<td>Any Decomposed</td>
<td>high medium</td>
<td>medium medium</td>
<td>high medium</td>
<td>medium high</td>
<td>excellent</td>
<td></td>
</tr>
<tr>
<td>Multiple</td>
<td>medium low</td>
<td>medium medium</td>
<td>medium medium</td>
<td>medium high</td>
<td>good</td>
<td></td>
</tr>
<tr>
<td>Multiple Separating Exclusion</td>
<td>high high medium medium medium</td>
<td>high</td>
<td>high</td>
<td>medium</td>
<td>excellent</td>
<td></td>
</tr>
<tr>
<td>Unique</td>
<td>medium high</td>
<td>medium high</td>
<td>high</td>
<td>high</td>
<td>very good</td>
<td></td>
</tr>
<tr>
<td>First</td>
<td>high medium high medium</td>
<td>medium high</td>
<td>medium</td>
<td>high</td>
<td>very good</td>
<td></td>
</tr>
<tr>
<td>Any</td>
<td>high medium medium medium</td>
<td>medium high</td>
<td>medium</td>
<td>high</td>
<td>good</td>
<td></td>
</tr>
<tr>
<td>Unique with default</td>
<td>medium medium high medium medium</td>
<td>high</td>
<td>medium</td>
<td>medium</td>
<td>good</td>
<td></td>
</tr>
<tr>
<td>Incomplete with default</td>
<td>medium medium medium medium medium</td>
<td>medium</td>
<td>medium</td>
<td>medium</td>
<td>good</td>
<td></td>
</tr>
<tr>
<td>Compact</td>
<td>medium low low low low low</td>
<td>low</td>
<td>low</td>
<td>low</td>
<td>low</td>
<td></td>
</tr>
<tr>
<td>More Compact</td>
<td>medium low low low low low</td>
<td>low</td>
<td>low</td>
<td>low</td>
<td>low</td>
<td></td>
</tr>
<tr>
<td>First</td>
<td>high medium medium medium</td>
<td>high</td>
<td>medium</td>
<td>medium</td>
<td>good</td>
<td></td>
</tr>
<tr>
<td>Unique</td>
<td>medium medium high medium</td>
<td>medium</td>
<td>medium</td>
<td>medium</td>
<td>good</td>
<td></td>
</tr>
<tr>
<td>First Decomposed</td>
<td>medium medium high medium</td>
<td>medium</td>
<td>medium low</td>
<td>medium</td>
<td>good</td>
<td></td>
</tr>
</tbody>
</table>

---

### Note 1: DMN Power

* All this is possible in DMN, in a standardized way, and immediately executable from the model.

* With all these possibilities, it is important to understand and exchange each other’s models. So, follow the DMN conventions.

* Methodology should not be blind for objectives.
**Model transformations and optimizations**

Lots of conversions/transformations/optimizations already exist:

* From tables to optimal code (Codasyl 1982)
* Lifecycle of hit policies (1988)
* Normalization of decision tables (1993)
* Decision models and business processes (2007)
* Factoring/Defactoring of decision models (1996)
* Decision model mining (2015)
* Mixed-paradigm process modeling (2016)
* Verification of tables (1998)
* Verification between tables (1998)
* From rules to tables (1982, 1993)
* From tables to minimal rules (1986)
* From logs to decision requirements (2010)
* Mixed-paradigm mining (2015)
* Construction methodologies (1986)
* Decision model dependencies (2012)
* Rule set consistency (2007)
* From Knowledge discovery to decision tables (2001, 2005)

---

**Note 2: Full DMN usage**

There is more than the basic big decision: *John Doe is 50 and has 30 years of service, how much does he get?*

The model should also allow to answer:

* What if I turn 45?
* Why do I get 27 days?
* What do I have to change to obtain x days?
* How do I get the maximum? (optimization)
* I have incomplete data, what can I already decide/exclude
* ...
*Thank you
and thanks to all who submitted their model*