Quick Guide to 
Decision Modeling 
using DMN 1.0

Gero Decker, Signavio, Inc.
Tom Debevoise, Signavio, Inc.

A White Paper
February 2017, 
2nd Edition
Table of Contents

1. Why read this Quick Guide? ................................................................. 3
2. Getting Started with DMN 1.0............................................................... 4
3. DMN Diagrams: Focusing on the questions to be answered ............... 8
   3.1 From Process Models to Decision Models ....................................... 8
   3.2 Divide & Conquer ........................................................................... 9
   3.3 When to divide Decisions into Sub-decisions .................................. 10
   3.4 Naming Conventions for Decisions.................................................. 11
   3.5 Inputs: Simple Types vs. Complex Objects (Business Objects) ........ 12
   3.6 When to use Knowledge Sources.................................................... 13
   3.7 When to use Business Knowledge Models ..................................... 13
4. Working with Decision Tables .............................................................. 14
   4.1 Complete or Incomplete? .............................................................. 15
   4.2 Choosing the right Hit Policy .......................................................... 15
   4.3 From Decision Trees to Decision Tables ........................................ 17
   4.4 From Check Lists to Decision Tables: The case of Unique vs. Any vs. First..... 18
   4.5 From Input Objects to Input Expressions ....................................... 20
5. Conclusion .......................................................................................... 21
6. Contact ............................................................................................... 22
1. Why read this Quick Guide?

There are many reasons why Business Decision Management (BDM) should be high on business leader’s objectives:

› The availability of big data and powerful analytics technology that uncovers new opportunities for “smarter decisions” and

› Complex requirements for compliance require equally complex decisions that are full traceable from business requirements to IT implementations.

To reap the strategic benefits of big data analytics, businesses need to connect the output to the operational decisions that drive processes. To make the connections businesses must have a deep understanding of these decisions and the logic behind them, the data that drives them and the processes they direct. BDM is a new discipline that identifies catalogues and models decisions, particularly operational decisions, in the enterprise.

The objective of a decision model is to create a shared understanding about its driving factors: the logic, data and processes. The recently approved Decision Modeling and Notation (DMN) 1.0 standard from the Object Management Group (OMG) is a notation that accomplishes these goals.

DMN 1.0 helps you to

› discover and optimize operational decisions in your organization

› document business logic for regulatory compliance

› design decision logic for rules-centric automation projects

› drive performance using analytics

The DMN standard also provides a powerful meta-model, notation and semantics for decision modeling. However, it also offers little guidance. Like most standards, there are no best practices or usage styles.

This Quick Guide serves as an introduction to decision modeling in DMN for novice modelers. It aims for guidance, not for completeness. It focuses on frequently asked questions and offers a variety of practical recommendations so that you can immediately start using DMN.

Happy decision modeling!
2. Getting Started with DMN 1.0

DMN is a critical addition for the business analysts that wish to model and improve the decisions that their businesses make. Prior to the DMN standard, there were no open standards for modeling the data and logic associated with the processes. In this quick-start guide, we will learn how to model decisions by immediately looking at a use case:

Imagine you are working at a bank. A customer makes a request to withdraw $9,000 from his savings account. Can you accept his request, process the transaction and hand him the money?

You have to make a business decision.

What are the criteria for the correct decision and what information is your choice based on? Is it relevant that the customer has withdrawn a large sum last week? Is there an upper bound?

The bank has standard rules for handling the request:

› There need to be sufficient funds in the account.

› The identity of the customer has to be properly verified. Certain documents must be present for transactions greater than $5,000. Moreover, which type of document is allowed depends on the customer's nationality.

Yet, these rules are just a portion of the logic needed to make a decision about the withdrawal of funds. Businesses make many different decisions that take place at the three levels of business:

› Strategic, these decisions have long-range impacts. They might include entering new lines of business, acquisition of companies and assets

› Tactical, these decisions involve existing lines of business. Tactical decisions include go-to-market plans, responding to immediate opportunities and taking advantage of short-term market advantages. Tactical decisions occur more frequently than strategic decisions

› Operational decisions. Operational decisions are 'baked' into existing business processes and practices. Operational decisions occur very frequently, sometimes scores of thousands each day. Examples include the prices of items, the discounts offered, and basic business decisions.

So the withdrawal decision is a simple example of an operational decision. It is connected to a process that validates and tracks the withdrawal of funds from customer accounts. Based on input facts, a decision outcome is derived, leading to an action in a process. Figure 1 presents a DMN model of this decision. DMN is an international standard and version 1.0 was released by OMG in Dec 2014.
DMN distinguishes between two levels: decision requirements diagrams (we will simply call them DMN diagrams) and decision logic.

Decisions are represented by rectangles and input data by rounded rectangles. In this example “Approve money withdrawal” is the top-level decision. “Sufficient funds” and “Valid identification” are sub decisions, the outcome of which is considered by top-level decision. They are also sub decisions because the questions they pose must be answered before the main question, “Approve money withdrawal”, is answered. The arrows depict a flow of the decisions into the top-level decision. There can also be more than one top-level decision.

Third element types, called knowledge sources help us depict dependencies on policies or regulatory requirements. In this case, the latter two sub decisions depend on “§154 AO”, a particular retail banking regulation.

DMN diagrams provide a high-level overview of how a decision depends on other decisions, policies or regulation and input data. Decision logic specifies the details of one decision, so that it can be easily and unambiguously understood by a human. Decision models can also be executable and included system components.

As shown in the process diagram below, decisions can be tightly connected to business processes. Operational decisions can affect a process in a number of distinctive ways. Decisions can direct the next step or activity to be taken, who is responsible for the activity, the course of action to be taken and the data required to complete the actions of the decision.
A decision is the act of settling a question or making a determination, from a number of input values, using logic defining how the output is determined from the inputs.

Decisions can depend on other decisions (sub-decisions).

Decisions depend on input data for determining the output. We will use the term "input objects" in this document.

Knowledge sources are authorities for decisions. Often, it is source documents, such as internal policies or external regulation that were used to derive the decision logic.

Information Requirements show the need input of information from either a sub-decisions or input data.
A DMN diagram does not depict the decision logic of a particular decision. This is done through either logical expressions or the decision tables that are attached to the decision. The decision table for “Approve money withdrawal” is shown in figure 2.

<table>
<thead>
<tr>
<th>AC</th>
<th>Sufficient funds</th>
<th>Valid Identification</th>
<th>Withdrawal approval</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Sufficient</td>
<td>Valid</td>
<td>Approved</td>
</tr>
<tr>
<td>2</td>
<td>Insufficient</td>
<td>Valid</td>
<td>Not approved</td>
</tr>
<tr>
<td>3</td>
<td>Insufficient</td>
<td>Invalid</td>
<td>Not approved</td>
</tr>
</tbody>
</table>

Figure 2 Decision table for “Approve money withdrawal”

The AC column orders the rules in the table. Under certain conditions, this order is important for determining which rule ‘fires’. The two columns on the left are the inputs. The column on the right is the output of the decision. Each row is a rule: It shows which combination of input values leads to which output value.

In this example, money withdrawal is only approved if there are sufficient funds and there is valid identification.

The rule that no identity document is required for withdrawals of less than $5,000 is part of the “Valid identification” decision. The empty cells denote a wildcard match for any value that is provided. The corresponding decision table is shown in figure 3.

<table>
<thead>
<tr>
<th>UC</th>
<th>Requested amount</th>
<th>Identity document</th>
<th>Nationality</th>
<th>Valid identification</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>&lt; 5000</td>
<td></td>
<td>(DE, CH, FR, US, UK)</td>
<td>Valid</td>
</tr>
<tr>
<td>2</td>
<td>≥ 5000</td>
<td>Passport</td>
<td>(DE, CH, FR, US, UK)</td>
<td>Valid</td>
</tr>
<tr>
<td>3</td>
<td>≥ 5000</td>
<td>EU identity card</td>
<td>∈ (DE, FR, UK)</td>
<td>Valid</td>
</tr>
<tr>
<td>4</td>
<td>≥ 5000</td>
<td>EU identity card</td>
<td>≠ (DE, FR, UK)</td>
<td>Invalid</td>
</tr>
<tr>
<td>5</td>
<td>≥ 5000</td>
<td>None</td>
<td></td>
<td>Invalid</td>
</tr>
</tbody>
</table>

Figure 3 Decision table for “Valid identification”
3. DMN Diagrams: Focusing on the questions to be answered

Every decision resolves or settles a question. Let's consider the money withdrawal example again. Here, the question is: Is the customer allowed to withdraw the requested amount from his savings account? The two possible answers are: “Approved” and “Not approved”.

Questions and answers are not graphically shown in DMN diagrams. They are attributes of decisions.

3.1 From Process Models to Decision Models

Business process models are a great way to show the decision response. Most decisions respond to and direct a business process. Here is another example from the financial sector: When developing a business process, the objectives of the story will suggest the correct question for the decisions. Consider the following story:

When an investment adviser enters into business or expands his operations, he must register with different authorities. Especially large advisers often register with the Securities and Exchange Commission (SEC), while others only register on the state level or even not at all.

![BPMN diagram](image)

The label of the gateway directly refers to the output of the decision. The branching conditions attached to the sequence flow edges describe the four possible answers.
Question: Where must the adviser register?

Process responses (Answers):
- State law
- State law, SEC potential
- None required

Therefore, the process objective determines the question to be answered. If the process was structured differently, the question would have looked different, too. For instance, alternative questions could have been "Do I need to register?" (Answer: Yes / No) or "Which states do I have to register with?" (Answer: list of corresponding states).

3.2 Divide & Conquer

Always focusing on the question also helps to subdivide complex decisions into sub decisions. Figure 5 shows a DMN diagram for the example.

The question whether the Adviser Act is applicable to the adviser has a large impact on the registration requirement: If the adviser does not qualify according to Section 202 then no registration is required at all. Therefore, “Applicability of Section 202 (a)(11) Adviser Act” was introduced as a sub decision.
The size category of the advisor is critical to the top decision. The size classifies the category and different rules apply to the adviser. Where the adviser has to register is split into the three sub decisions:

- “Registration requirement (small adviser)”,
- “Registration requirement (medium-sized adviser)”
- “Registration requirement (large adviser)”.

Slicing up a larger scenario into smaller pieces, which can then be treated (“managed”) individually, is a common modeling pattern that is not only found in decision modeling: Divide & conquer.

You can apply divide and conquer when moving from the top-level decision to sub decisions (vertical divide & conquer). A good example is the sub-decision “Applicability of Section 202 (a)(11) Adviser Act”. Divide & conquer can also be used in segmentation (horizontal divide & conquer). Slicing up the decision according to the size classification is an example.

3.3 When to divide Decisions into Sub-decisions

Finding the correct level of abstraction is an important success factor in every modeling endeavor. This is especially true when deciding to use one “bigger” DMN decision or rather several “small” DMN.

Consider the investment adviser DMN diagram in figure 5. Here, the different decisions are small enough to define the decision logic at a latter point. Also, the decisions are big enough not to get lost in huge diagrams. Here are some hints that can help you decide when to split up decisions into sub decisions:

<table>
<thead>
<tr>
<th>Criterion</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Necessity</td>
<td>Sub decisions are often necessitated by the objectives of the to-level decision.</td>
</tr>
<tr>
<td></td>
<td>If in the example, different size-specific registration requirements needed to be decided before the top-level decision could be made. Simply put, most business decisions need the input of sub decisions.</td>
</tr>
<tr>
<td>Reusability</td>
<td>Sub decisions can be reused in different decision models and different contexts.</td>
</tr>
<tr>
<td></td>
<td>For instance, the “Investment adviser size classification” might not only be relevant for the question registering an adviser but also for other contexts.</td>
</tr>
<tr>
<td>Complexity</td>
<td>Once a decision has more than 7 inputs and/or sub decisions, this is a clear sign that the decision logic will most likely be very complex. The idea is that humans are still able to understand the decision logic – and not only machines.</td>
</tr>
<tr>
<td></td>
<td>So better split a very complex decision into two or more decisions.</td>
</tr>
<tr>
<td>Authority</td>
<td>DMN Knowledge Sources can be used to depict policies or regulatory requirements that influence / guide a decision. To show that part of the decision making is based on an internal policy and another depends on some external regulation, could be a good reason for splitting up the decision into two decisions.</td>
</tr>
</tbody>
</table>
Decision maker

Who is making the decision? Is it a person or is it a machine? If different people or different systems are making different parts of a decision, then you might need to split up the decision into decisions made by one machine or one role only.

“Decision maker” is an attribute of decisions in DMN diagrams (no graphical representation).

Decision owner

Every decision can have a decision owner. This is the person responsible for the decision outcome. If there are different parts of a decision which different owners, you might want to split the decision up according to ownership boundaries.

“Decision owner” is an attribute of decisions in DMN diagrams (not graphical representation).

Table 2 Criteria for splitting up decisions into sub decisions

3.4 Naming Conventions for Decisions

Working with decisions and sub decisions suggests a need for naming conventions in DMN. Table 2 shows the most popular labeling styles found in DMN diagrams.

<table>
<thead>
<tr>
<th>Naming style</th>
<th>Example</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Activity style</td>
<td>Determine where to register</td>
<td>The decision has the same label as the corresponding BPMN task. The act of decision making moves into the center.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This is a “verb object” style. Typical verbs are “check”, “determine” or “calculate”, “select”, “choose”.</td>
</tr>
<tr>
<td>Output style</td>
<td>Investment adviser size classification</td>
<td>The decision has the same label like the output variable of the decision logic.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Often, you can find words like “applicability”, “eligibility”, “score” or “ranking” as part of this labeling style.</td>
</tr>
</tbody>
</table>
### Question style

<table>
<thead>
<tr>
<th>Provides advice to registered investment companies only</th>
<th>The decision label equals the question or at least is an abbreviated version of the question.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This labeling style is often observed when turning complex / raw input data (such as customer base in this example) into something meaningful.</td>
</tr>
<tr>
<td></td>
<td>Attention: Questions are often very long and therefore lead to very long decision labels.</td>
</tr>
</tbody>
</table>

Table 3 Labeling styles for decisions

**Recommendation:** Use the activity style for the top-level decision, to highlight the connection to the surrounding process. Use the output style for all other decisions. In some cases, the question style is more intuitive than the output style.

### 3.5 Inputs: Simple Types vs. Complex Objects (Business Objects)

On the DMN diagram level, the modeler can select the level of granularity of input objects. From the graphical representation in the DMN diagram, we cannot tell whether an input object is a complex object or a simple type.

In the example above, “AUM” (which is the abbreviation for “Assets under Management”) is most likely a simple type, namely a USD amount. The case of “Location” is not so easy. It could either be a simple type (e.g. one of the US states) or a complex object consisting of a structured address, including street name, zip code, state, country or even geolocation. Some business analysts call these business objects.

Each complex object has multiple fields. Every field in turn can be a complex object or a simple type. Zip code and state are examples for simple type fields. Geolocation is an example for a complex object field: It consists of the latitude and longitude.

Many times, the decision requirements modelers might lack detailed understanding of the structure of the data. Therefore, they might not know whether it is a simple type or a complex object.

This is not a problem when drafting the first version of a decision model. In early phases of DMN diagram modeling, it is more important to identify the right questions and answers (the decisions) and their dependencies.

When defining the detailed decision logic, an understanding of the underlying data model is required. Decision modeling should be a collaborative effort between requirements modelers, people with deep understanding of the data model and subject matter experts who can explain intricate details of the decision logic.
Some organizations have a central taxonomy or data dictionary, which contains the most relevant terms / objects and their relationships. This is valuable input for decision modeling. If properly done, a data dictionary accelerates the selection of the right input objects and maps high-level concepts to technical objects in later stages.

**Recommendation:** Building a decision model is an iterative process. Do not wait for too many prerequisites (e.g. proper central taxonomy) to start modeling. Rather use a collaborative approach that involves the right people.

### 3.6 When to use Knowledge Sources

Many decisions are driven by external regulation and internal compliance policies. Early collection of the regulatory requirements and dependencies on policies early on helps create a meaningful decision model.

A question often asked whether to name the entire regulation / policy or to directly point to a specific chapter / section / paragraph within such a document. Organizations frequently connect their decisions logic with the proper regulations.

**Recommendation:** Use knowledge sources to list the relevant dependencies on internal policies or external regulation.

### 3.7 When to use Business Knowledge Models

Reuse of decisions is one of the strengths of DMN. Many DMN diagrams contain sub decisions which are specified in a different DMN diagram. The question / answers and the dependencies on sub decisions, knowledge sources, input objects and decision logic are inherited from the other DMN diagram.

DMN offers yet another reusability mechanism, which we have not talked about so far. The “business knowledge model” represents reusable decision logic.

Business Knowledge is used where two decisions share the same decision logic, but depend on different sub decisions and on different input objects, i.e. different variables. In this case, business knowledge models can be used to create a central definition of the decision logic which can then be “invoked” by two different decisions.

Reusable decisions (same input objects, same dependency on sub decisions, same decision logic) is common, reusable decision logic with different input objects / sub decisions is rare. The decision logic embedded in a decision model can be reusable across models. An alternate to the Business Knowledge is to connect to the embedded decision table.
4. Working with Decision Tables

A colleague of yours sends you an email about how discounts are calculated for customers:

- Adults who are not married get a 10% discount.
- If you are married, you get at least a 20% discount.
- If you have kids, you even get a 30% discount.

According to this description, the decision depends on the following inputs: Age (adult / child), marital status (married / not married), parental status (kids / no kids).

Decision tables are the notation of choice to specify decision logic in DMN. When directly translating the rules from the example, Figure 6 shows the resulting decision table.

When starting to develop the ‘rules’ or logic for these word problems, you should discern what inputs will settle the questions posed by the decision. In this case our question is ‘What discount should we give to our customer’. In our assignment of output, we can see that it should be possible to create a rule for each combination of the inputs. In this analysis we decide on the way the rules are evaluated in a decision table. For instance, there might be a restriction that only one row in the table will return an output result. This is known as a ‘Unique hit’ policy. That said, the team has not articulated all the rule combinations. As a rule analyst, you have the option of explicitly asking for the desired rule output for all the combinations, or you can have the colleagues develop the rules, based on an analysis of the written requirements. Frequently, the latter analysis is how organizations develop their logic.

![Figure 6 Decision table with errors: incompleteness and overlapping rules, this is a partial effort](image)

Where there is no entry in an input cell, there is an implicit ‘wild card’ that matches any value. For instance, rule 2 matches any person with a marital status of “Married”. There are several errors or overlaps in this decision logic. The decision logic verification functionality of Signavio identifies the following problems:

- The following combinations of input values are not covered by any rule (Age = Child, Marital status = Not married, Parental status = No kids)
- Unique hit policy violated for Rule 1 and 3, these combinations of input values are covered by more than one rule: (Age = Adult, Marital status = Not married, Parental status = Kids)
- Unique hit policy violated for Rule 2 and 3, these combinations of input values are covered by more than one rule: (Marital status = Married, Parental status = Kids)
Because we are seeking to create a ‘unique’ policy for the rules in the tables, these statements suggest this decision table is “incomplete” (error 1). Also, there are input combinations that are covered by several rules in the decision table (errors 2 and 3).

Decision tables are not required to be complete or gap free: In a discount calculation example, an input value combination that is not covered by any rule could easily be interpreted as “no discount”. Overlapping rules are permitted, if they all have the same output, or if there is a certain priority configured for which of the matching rules actually applies. You can configure a DMN decision table regarding completeness and hit policy.

The hit policy in a DMN Decision Table specifies how the result of the decision table will be evaluated where there are multiple matches for a given set of inputs. The hit policy contains additional information that can be used to check correctness of decision table at design-time.

4.1 Complete or Incomplete?

In our example, there is no rule for children who are not married and who do not have kids. According to the DMN standard, this is an “incomplete” decision table. The simple resolution for this example would be to add another rule for exactly this case. This would turn it into a complete decision table.

Completeness is only possible if the value domains for all inputs are known. In our example, this is the case: each input has a value domain consisting of two values.

**Recommendation:** Use complete decision tables, whenever possible. This avoids misunderstandings about the decision logic.

4.2 Choosing the right Hit Policy

Hit policies and the problem we are solving determine the semantics of a decision table. They also impose requirements on how rules must be structured. The conditions of the business rules also can dictate the hit policy and the rules within the table.

<table>
<thead>
<tr>
<th>Hit policy</th>
<th>Description</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Unique (single)</strong></td>
<td>Only one rule is allowed to fire for any one combination of inputs.</td>
<td>Where a uniform policy is required for the combination of inputs</td>
</tr>
<tr>
<td></td>
<td>In this context, all inputs are assumed to be independently partitioned from each other, i.e. that any combination is actually possible. For our example, it is assumed that children can also be married.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Our example decision table violates the Unique requirement, as there are rules that overlap.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Unique (single) is the default in the DMN standard.</td>
<td></td>
</tr>
</tbody>
</table>
Any (single) | Multiple rules cover the same combination of input values. However, this overlap is only allowed if the overlapping rules have the same output. A diagnosis of a medical condition of the or the output of lab results might need an Any’ policy if various ranges of similar diagnostic test identify the same condition.  

Figure 2 shows a valid example of a decision table with hit policy Any. 

Priority (single) | Rules can overlap with different outputs.  
In order to determine the resulting output, outputs are ordered according to priority (e.g. the highest discount wins).  
Priority rules resolve ‘race’ conditions, such as evaluating the weighted scores of various priorities. 

First (single) | Rules might overlap. But only one rule fires: The First hit policy simply assumes an ordering of the rules – they are evaluated from top to bottom. As soon as one rule fires, the output of that rule is used as result of the decision. Once one rule hits, no other rule is checked.  
When interpreting our example decision table using the first hit semantics, adults who are not married but have kids, would only get 10% discount (instead of 30%). 

No Order (multi) | Multiple rules can fire without an explicit order. Multi hit tables either return a set or aggregate the final results of the decisions. DMN’s default is provide the set of output values as result. In our example this would be (10%,30%) for not married adults with kids. Multi-rule table can apply aggregation functions such as min, max, avg to derive a single value from a set of output values.  
Multi hit policies are good for score cards that accumulate for matching inputs. For instance the condition of an inspected item might be based on multiple checks and with points off for various defects.

Table 4 the most relevant DMN hit policies

All of these hit policies (except for the First hit policy) are declarative in nature, i.e. the ordering of the rules does not have any semantics. The First hit policy is not declarative, but procedural. This hit policy might be used by people with a technical background (where if then else statements are also evaluated in a particular order), but it is counter to the declarative nature of DMN.

Recommendation: If you are unsure of the hit policy, start with the Unique hit policy. This forces you to define rules that do not overlap each other. As shown above, some applications need overlapping and additive rules. Make certain the rules specify the desired results. In some cases, the “Any” hit policy leads to more compact and more easily understandable decision tables. Do not use the “First hit” policy and try to avoid the Priority hit policy. Multi hit is typically only necessary for certain scoring and calculation scenarios.
4.3 From Decision Trees to Decision Tables

Decision tables are equivalent to decision graphs (and vice versa). It is almost as common to use decision trees to describing decision logic as it is to use work flow diagrams.

You return to the team to discuss your findings regarding incompleteness and the missing rules. Obviously, there has been some misunderstanding. To resolve this, your colleague draws a small flowchart to clarify the semantics of the discount calculation.

![BPMN diagram used as decision tree](image)

**Figure 7:** BPMN diagram used as decision tree

BPMN was used to draw this decision tree. BPMN is not meant to be used for modeling decision logic. An upcoming version of DMN will most likely include graphical decision trees as an alternative notation of decision logic.

Decision trees can very easily be translated into decision tables. Each decision node from a decision tree is translated into a column in the decision table. Then, the table is filled from left to right.
When a decision tree is translated into a decision table, the resulting decision table automatically fulfills the Uniqueness requirement. The completeness requirement is also fulfilled if each decision node of the tree covers the complete value domain of the corresponding input.

**Recommendation:** Do not encode decision logic in BPMN diagrams. Use DMN decision tables instead.

### 4.4 From Check Lists to Decision Tables: The case of Unique vs. Any vs. First

Let’s assume that a purchase order can be accepted if

- a valid quotation number is referenced,
- the pricing matches the quote,
- all required items are included in the order, and
- the quotation has not expired yet.

The decision logic behind this check list is: If all conditions are met, then the purchase order is “acceptable”. If at least one condition is not met, the purchase order is “not acceptable”.

Translating this example using first hit semantics leads to the decision table in figure 9.

![Decision table example](image-url)
This example can easily be translated into an Any hit decision table and a Unique hit decision table, which is illustrated in figures 10 and 11.

<table>
<thead>
<tr>
<th>AC</th>
<th>Valid quotation number</th>
<th>Matching price</th>
<th>All items included</th>
<th>Quotation expired</th>
<th>Acceptable?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>2</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>3</td>
<td>true</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>4</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>5</td>
<td>true</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>true</td>
</tr>
</tbody>
</table>

Figure 10 Any hit decision table derived from check list

<table>
<thead>
<tr>
<th>UC</th>
<th>Valid quotation number</th>
<th>Matching price</th>
<th>All items included</th>
<th>Quotation expired</th>
<th>Acceptable?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>2</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>3</td>
<td>true</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>4</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>5</td>
<td>true</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>true</td>
</tr>
</tbody>
</table>

Figure 11 Unique hit decision table derived from check list

The big difference between the First hit table and the other two is that the First hit table is considerably smaller. The second rule fires for all input combinations, serving as a “catch all” rule.

The Any hit table has one rule for every input, covering all negative cases for that input.

The Unique hit table has a structure like if it was derived from a decision tree (see above). The requirement that the rules must not overlap, makes the decision table appear more complicated than the Any hit decision table.

In general, decision tables are not the most intuitive way of representing a simple check list. Decision tables are the business user-friendly option we have in DMN.

**Recommendation:** Use “Any” hit tables to represent check list-like scenarios.
4.5 From Input Objects to Input Expressions

In the previous section we have briefly touched on the question of simple types vs. complex objects. When relating DMN diagrams to decision tables, this is one of the central questions.

Inputs to decision tables must always be a simple type. Table 5 shows a list of typical examples for simple types.

<table>
<thead>
<tr>
<th>Type</th>
<th>Example</th>
<th>Operators</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Valid identification = {true, false}</td>
<td>= (equals)</td>
<td>True or false.</td>
</tr>
<tr>
<td>Enumeration of values</td>
<td>Currency of Trade = {EUR, USD, CHF, GBP, ...}</td>
<td>= (equals) ≠ (not equals) ∈ (element of) ∉ (not element of)</td>
<td>List of values.</td>
</tr>
<tr>
<td>Number</td>
<td>Requested amount = USD dollar amount with 2 decimals</td>
<td>= (equals) &lt; &gt; (less, greater) ∈ (in range)</td>
<td>Number with unit.</td>
</tr>
<tr>
<td>Text (String)</td>
<td>Product ID</td>
<td>∋ (contains) ∋ (not contains) <em>T (starts with) T</em> (ends with)</td>
<td>Mostly used for data acceptance / validation testing.</td>
</tr>
</tbody>
</table>

Table 5 Simple types

As discussed above, input objects in DMN diagrams can either be simple types or complex objects. In the case of simple types, the input value is passed directly into the decision table (e.g. AUM). In the case of a complex object, one or many fields must be chosen which are simple types (e.g. Customer.age, Location.state).

Sometimes, complex objects map to technical objects (e.g. Java class Customer). In other cases, complex objects are only a convenient grouping for business level-modeling and only the fields are actually mapped to technical data structures.

When developing business rules, it is best to use as many well defined lists (enumerations) and number ranges. Lists and number ranges can provide a wide range of choice in detailing business rules. In addition these rules can be checked for uniqueness and completeness.

**Recommendation:** Avoid using simple type Text, as it does not allow for sophisticated completeness and overlapping rules analysis. Apply enumerations and numbers as much as possible.
5. Conclusion

This “quick start” set out the core elements of the Decision Model and Notation standard—decisions, input data, knowledge sources, and business knowledge models. These elements can be combined into decision requirements diagrams to describe the decision making.

Decision requirements diagrams can specify how decisions in a process should be made. They can also specify the logics requirements for decision automation using a decision table. The DRD breaks down decision making into its component pieces for clarity. They show what information is required for each part of the decision, as well as where the rules, guidelines, policies, and know-how to make the decision correctly can be found.

Decision modeling is a relatively new approach for most organizations. This white paper provides the tools for users to read decision models and represent decision models in a standard way as decision modeling is developed as a discipline alongside process modeling. Knowing the notation for decision models is necessary for decision modeling, but just as with process modeling, the notation is not a methodology, and developing effective models will likely take practice and training.

We also set out the logic elements of a decision model, especially the decision table. Logic elements can be combined into decision requirements diagrams to describe precisely how a decision can be executed. That is, diagrams can be extended with decision logic to form a complete definition of exactly how the business wishes to make a decision.

The Signavio Process Manager supports the DMN modeling described here. It creates a comprehensive environment for collaborating on the discovery, management and improvement of your decisions. With the Process Manager, decisions can be reused across other decisions and they can even be connected with the processes they drive. Process Manager empowers the people in your organization that must manage even the most complex decision.
6. Contact

Publisher
Signavio GmbH
Kurfürstenstraße 111
10787 Berlin
Germany

+49 30 856 23 54-0
+49 30 856 23 54-19

info@signavio.com
www.signavio.com