MODELING DECISIONS WITH IBM OPERATIONAL DECISION MANAGER

AN AUTOMATION AND MODELING PRIMER FOR BUSINESS EXPERTS

Operational decisions are fundamental to your organization, and they're everywhere! Just think of the thousands of decisions made in day-to-day operations—managing risk, assessing credit and loan applications, adjusting pricing, offering promotions to loyal clients, detecting fraud—you name it; there are operational decisions behind pretty much every aspect of your organization.

Operational decisions stem from business policies, which evolve as your organization adapts to changes in its environment, like changes in regulations, new promotional campaigns, customer support initiatives, new competitors, and so on.

Naturally, the choices you make directly affect your organization's results. Operational decisions are part of this picture too. When circumstances require agility and efficiency for immediate action, your business experts' ability to quickly adjust how operational decisions are made can make or break the organization.

Decision modeling makes this agility available to business experts.

Modeling operational decisions

Often operational decisions are made in the code of enterprise applications, which makes it difficult to access and modify them. When the policies that govern the operational decisions are expressed in code, updating them is not only costly but it also requires specific IT expertise. Even the smallest tweak (changing a minimum credit score from 600 to 620, say) can take months to come online. And by that time things might have changed again.

Decision modeling can help you loosen this Gordian knot, and maybe slice right through it. It is a straightforward and code-free way to express—and refine—your operational decisions through a structured, visual representation of a decision. Using this approach, business experts and IT specialists alike can model operational decisions by specifying:

- the information you use to make decisions
- how to make the final decision with that information—the decision logic

With decision modeling you have a direct, hands-on way to describe operational decisions in a form that can be used right away by your enterprise applications, replacing the traditional code-based implementation. That is, you regain control over the decision logic, and when that logic goes live.

Imagine you want your online shop to apply a discount to customers. The underlying application applies the discount—the operational decision—according to the decision logic that that you define in your decision model. If you want to add extra discounts for a new promotional campaign, you just edit the related discount decision, and as soon as you've deployed your updates the application applies discounts in line with the new campaign.

Decomposing decisions

Basically, modeling a decision involves breaking it up into a series of smaller decisions.

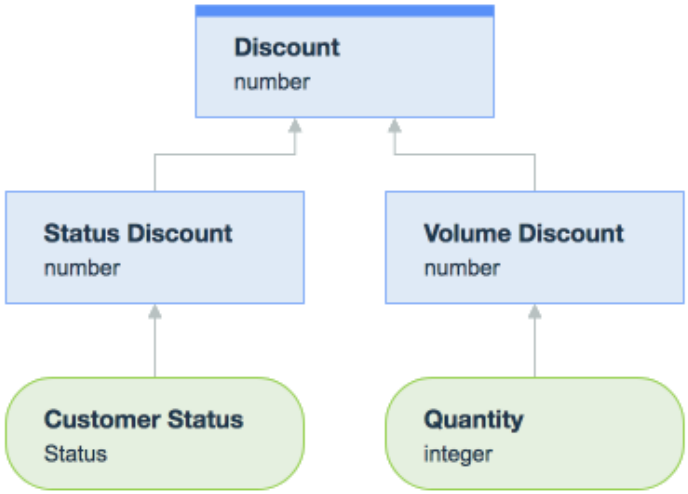
If our online shop offers discounts depending on things like the number of items in the basket, the total undiscounted price, the customer's loyalty, the time of the day, and so on, each of these factors can be used to make simple decisions that all together make the overall discount decision. The customer loyalty discount—5% off for Bronze members, 10% off for Silver, and so on—is a smaller decision that contributes to the result of the final decision, which takes all the factors into account.

Decomposing a decision into smaller elements isolates its building blocks and shows how they relate to each other. It makes it easier to understand the logic at stake, define it in a straightforward way, and adjust it when things change. It's also an opportunity to avoid reinventing the wheel. Chances are that the smaller decisions can be aligned with policies already documented within your organization; the discounts for customers in the loyalty program is probably already defined in the loyalty program's charter.

As you break up a decision you reach a point where you need information to inform your decision. The customer's membership of the loyalty program is not a smaller sub-decision, it's information—input data—you need to make the decision. Input data can come from an external system, such as a client database.

Representing decisions graphically

The standard way to represent a decision model is as a decision model diagram, which lays out the decision points, called nodes, and the dependencies between them. It also shows the input data used in making the decision. The diagram below shows a simple discount decision.



From top to bottom:

- The overall *discount* **depends** on the *status discount* and the *volume discount*.

- The *status discount* **depends** on the *customer status*, and the *volume discount* **depends** on the *quantity* of items. The *customer status* and the *quantity* are what you need to know to make a discount decision.

The diagram is high-level and doesn't give actual values for its elements. It shows how the decision is made of simpler decisions and the data you need to make the decision.

Notice that each node has a type that tells us what kind of result we'll get from the node. For example, the discount sub-decisions, *status discount* and *volume discount*, give a number -- a percentage (such as 12% or 0.12). The input data is also typed: *quantity* is an integer (a whole number), and *customer status* comes from a list of the possible loyalty statuses (for example Bronze, Silver or Gold).

Each decision node gives a value that nodes further up the diagram can use. In our discount example, the final *discount* node specifies how to use the numbers from the *status discount* and the *volume discount* to compute the final discount.

Defining the decision logic

Once the high-level diagram is in place you can start to define how each node arrives at its result. Generally, you have two ways to do this: decision tables and business rules.

Decision Tables

Decision tables link conditions with the resulting actions. In the discount example, we could define the logic for the *status discount* node like this:

	Customer status	Status discount
1	bronze	0%
2	silver	5%
3	gold	15%

The *Customer status* column gives the condition, and the *Status discount* column gives the corresponding result for the node. If customer status is bronze, there is no status discount, if it's silver the status discount is 5%, and so on.

We can use a decision table for the *volume discount* node too, but here the condition is based on ranges of values:

	Quantity		Volume discount
	min	max	
1	0	1,500	0%
2	1,500	3,000	5%
3	3,000	10,000	10%
4	*Otherwise*		15%

If *quantity* is between 0 and 1500, no discount is applied, for quantities between 1,500 and 3,000, the volume discount is 5%, and so on.

In general, decision tables are best when you can list the all the cases and their corresponding values. And you can create more sophisticated decision tables by grouping multiple conditions and actions.

Business rules

When you can't list all the possible cases, or your policies are more complex, you use business rules.

In a nutshell, a business rule is a policy statement written in "an almost natural language". It describes how to get the result of a node using the available data, which might include the results of other nodes.

In our example, the final *Discount* decision is where you bring together the status and volume discounts. One way to it would be to just add them:

```
set decision to the volume discount + the status discount
```

But your policy might be to offer the highest of the two discounts, so you could write it like this:

```
if the status discount is more than the volume discount then
  set decision to the status discount
else
  set decision to the volume discount
```

Of course, there are many ways to combine the discounts to get the final discount. Business rules give you the flexibility to express the logic of your decision just how you want.

Decision modeling at your finger tips

The Operational Decision Manager product family provides a fully-fledged decision modeling environment for business experts, to help them take control of the decision-making process. Designing decisions, defining the driving business logic, and validating them is just a few steps and clicks away.

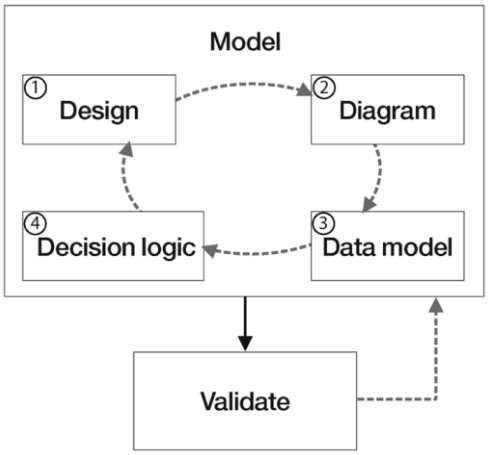
With decision modeling in Operational Decision Manager you have a powerful and easy-to-use tool to:

- Visually represent the structure of an operational decision, its parts, and their dependencies. This is where, you break down your operational decision into more manageable intermediate decisions.
- Define the underlying business data model—the kinds of things involved in making the decision. In our discount example the client, the client status, the product, and the quantity are types of business data. They aren't automatically available, because they are specific to the business domain.
- Author the rules used to arrive at the decision.

And of course, once that's done Operational Decision Manager includes features for you to validate your decisions right away and deploy them into a decision service for production and automation.

Walking through decision modeling

Decision modeling takes you smoothly through the typical modeling cycle in an iterative way.



Design

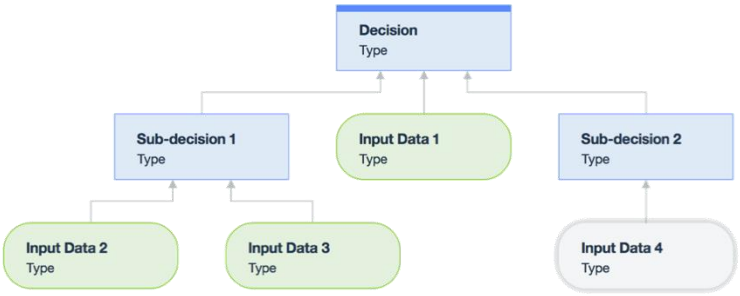
Before starting to model a decision, it's a good idea to think about what you would like to achieve: What is the actual decision you want to make? Is this decision dependent on other decisions? What is the business data that is involved in the decision? What data influences the outcome? This is the design part that you then translate into a visual representation.

Diagram

A decision model diagram gives you a high-level representation of how decisions and the data required to make are structured and related to each other.

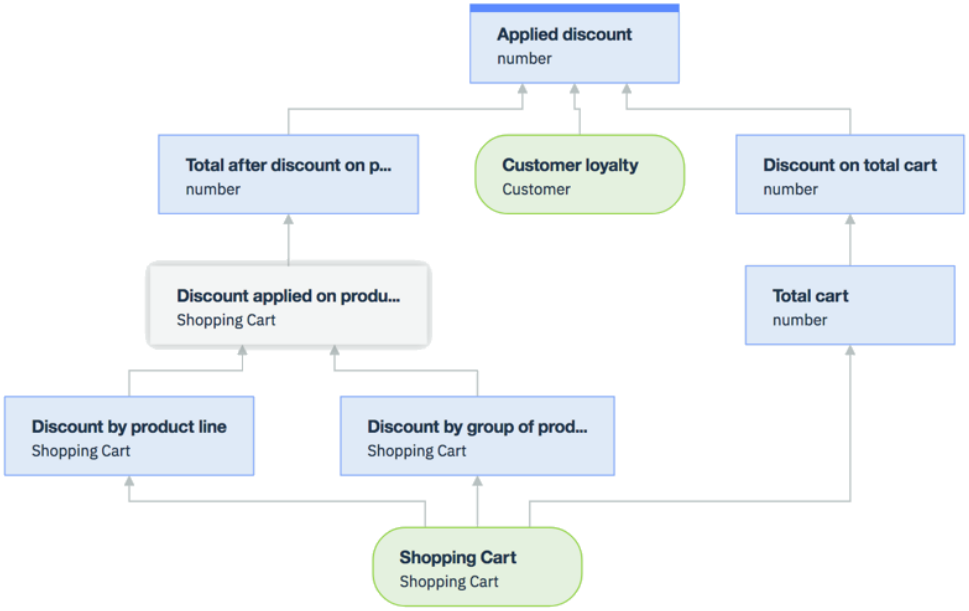
In Operational Decision Manager decisions are represented by blue square boxes, called **decision nodes**, and input data are show as green rounded boxes, called **data nodes**.

In the empty diagram below the final decision, with a thick top edge, is broken down into two intermediate decisions. The pointing arrows that connect the final decision to the intermediate ones show the **dependency relationship** between the decisions and data. The arrows also give the direction of the data flow in the model. The outcome of the final decision depends on the outcome of *sub-decision 1* and *sub-decision 2*. In turn, the outcome of *sub-decision 1* and *sub-decision 2* depends on the value of the input data. Input data influence the decision outcome and do not have any dependencies; they exist by themselves and can be referred to as primary input data.



The diagram below shows the model of a decision for the discount policy of a supermarket. The final decision answers the question:

"What is the total discount to apply to a customer based on the total of the shopping cart, the loyal customer status, and the promotional discount on certain products or group of products?"



The **Applied discount** depends on two other decisions and an input data:

- **Discount on total cart**: "*What is the discount to apply depending on the shopping cart amount?*"
- **Total after discount on products**: *"What is the total discount to apply for purchasing a product or a group of products in promotion?"*
- **Customer loyalty**: *What level has the customer reached in the loyalty program?*

The input to **Total after discount on product** is the outcome of an intermediate decision **Discount applied on products** that answers the question: *"What is the best discount to apply to a cart by product line or by quantity purchased?"*

And the input to **Discount on total cart** is the outcome of the intermediate decision **Total cart** that answers the question: *"What is the total of the shopping cart?"*

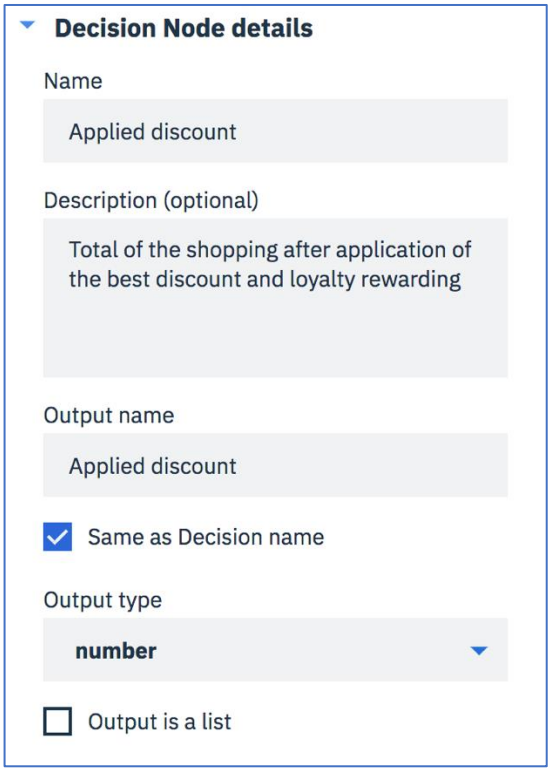
Here we have effectively decomposed each step of the decision into its building blocks. This process sheds lights on the structure of a decision and the chain of dependencies between its various components.

Customer loyalty and **Shopping cart** are primary **input data**, which means that they do not depend on external input to set their value.

Data model

The outcome of a decision always has a type. The type identifies what kind of thing we're dealing with. Is it a number (final price = 74.99)? Or a string of text (promotion message = "Congratulations, your 10th purchase is free!")? Or perhaps it's a true/false value called a Boolean (loan approved = true)?

There are simple types predefined in the application that you can be use as is. For example, in our discount case, the outcome of the **whole** decision is a discount, so its type is a number.



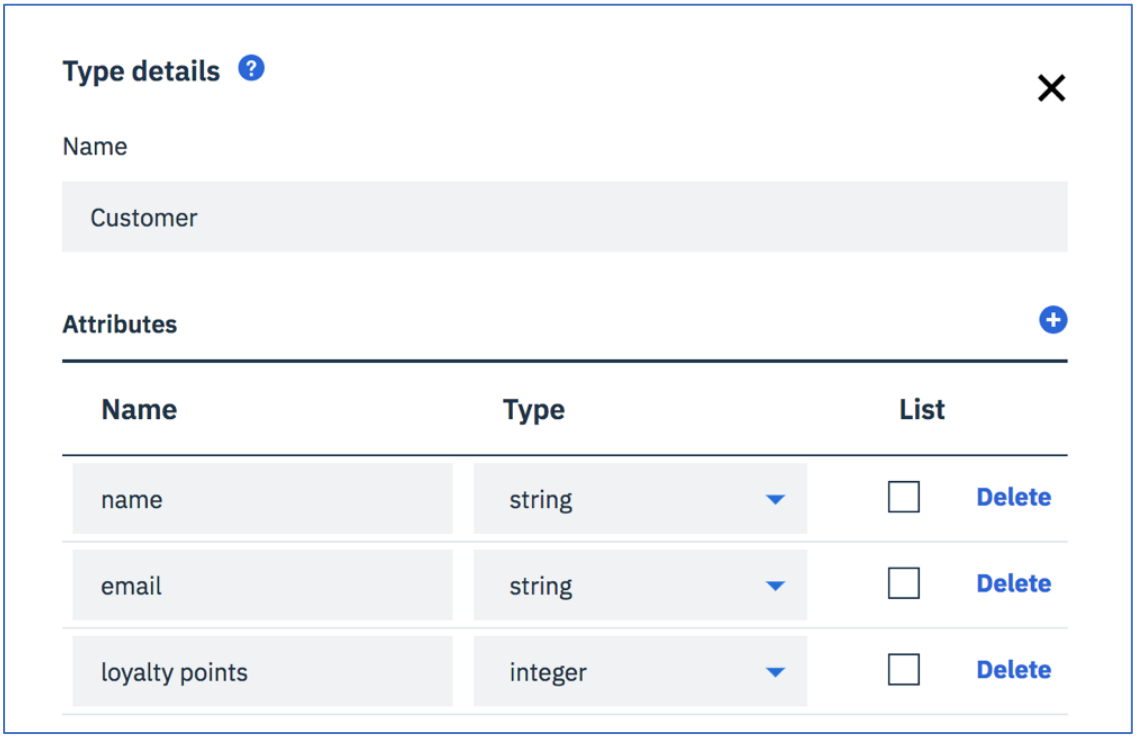
Of course, the world isn't always easy to divide into simple types, and the outcome of a decision can also be a more complex construct that you tailor to the needs of your business domain. These custom constructs correspond to business data.

In the same way that you model decisions, you also describe the entities, or data, that govern your business activity and their relationship in an abstract way using a model, referred to as a business data model.

For example, you can represent the typical activities of a supermarket with business data using your own language: customer, product, price, promotion, shopping cart—but also employee, supplier, stock, fleet of vehicles, and so on. These terms make up the data model vocabulary.

Because the data model vocabulary is specific to a given domain, decision modeling lets you create and control business data relevant to that domain, and the operational decision you are modeling, very easily using text fields, and drop-down menus.

The illustration below shows the panel that defines the **Customer** business data. The business data is defined by a name and a set of attributes that you control. Here the customer is associated with a name, an email address, and loyalty points.



Decision logic

At this stage in the cycle, we have high-level representations of a decision and its input data. Now we need to define exactly how each decision node processes its inputs to generate its result: the decision logic. Another way of thinking about decision logic is that it's where the policies of your organization influence the final decision.

You express the decision logic with **decision tables** and **business rules**.

Decision tables are like spreadsheets. The columns on the left are the conditions. They show the cases you're considering. The right-most column is the action you want to take for each case. In general, you group rules (policies) that have similar conditions and actions into one decision table where each row of a decision table corresponds to a rule.

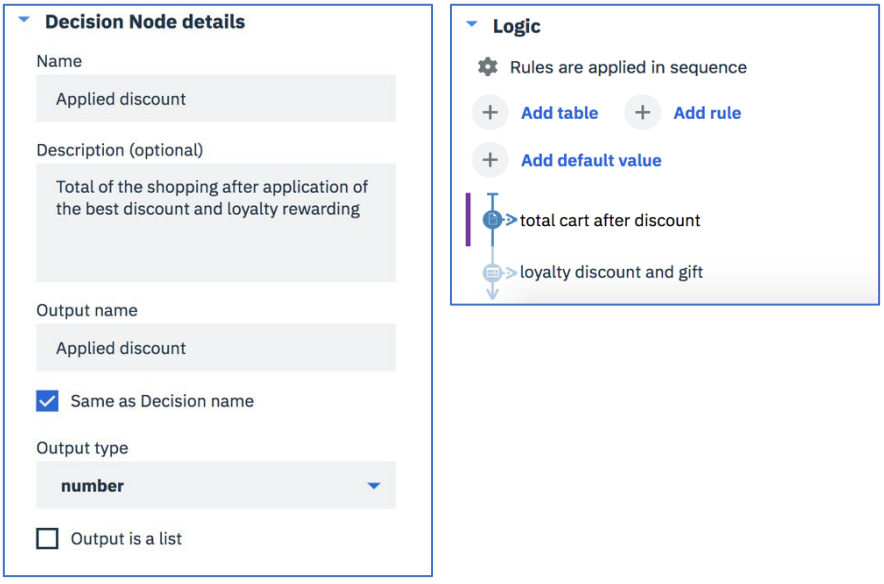
	Total cart		% of Discount on total cart
	min	max	
1	< 20		0
2	20	100	1
3	100	150	3
4	150	300	7
5	≥ 300		10

Grouping rules that have similar conditions and actions is not always possible. For these cases you can use business rules.

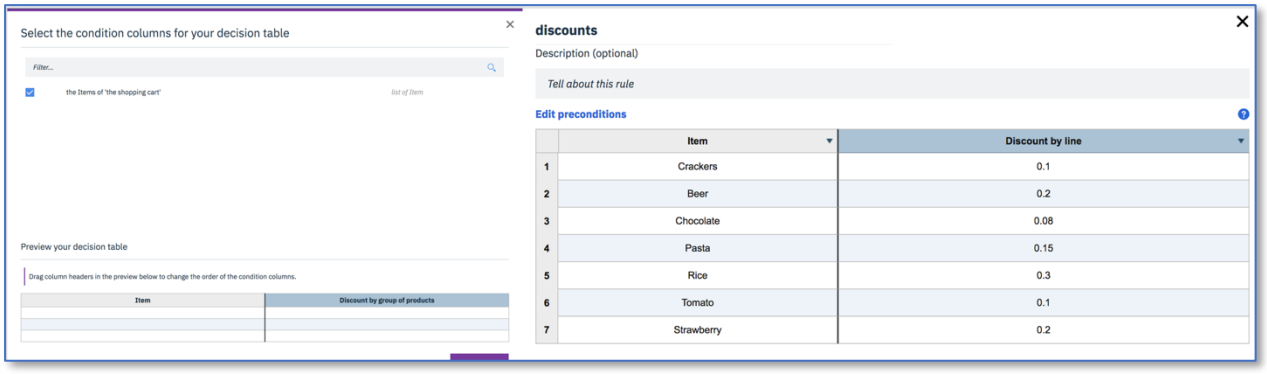
A business rule is a set of "if-then" phrases that you write in Business Action Language (BAL). BAL is very close to natural human language, so all users can work with them—not just those who know programming languages.

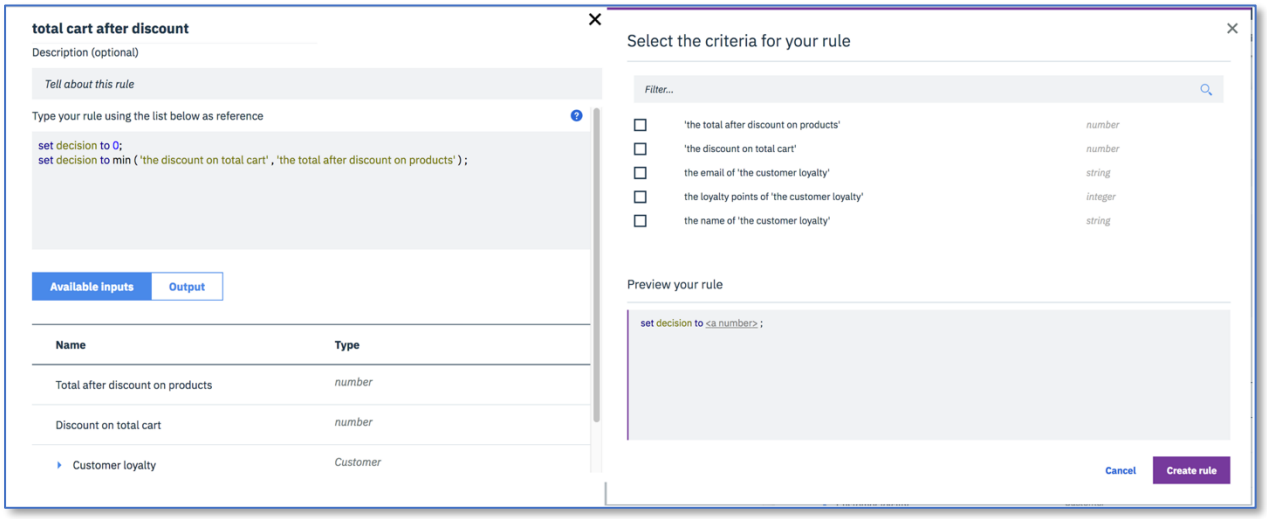
```
set decision to 0;
for each Item in the items of 'the shopping cart':
-set decision to decision + the unit price of this Item * the quantity of this Item;
```

You define the decision logic for each decision node directly in the decision modeling environment.



To create decision tables or author business rules, you choose from a list of statements that use terms (vocabulary) based on the terms in the decision and data models, then set their values so they can be validated, and later deployed to production.





With the Operational Decision Manager decision modeling environment, creating decision-based applications for rapid deployment has never been so simple and rewarding!

Now what?

Try the ease and power of decision modeling with IBM Decision Composer, the free, streamlined edition of Operational Decision Manager, and execute up to 1000 decisions monthly, or explore decision management more deeply with Operational Decision Manager for Developers—it's free too!