**Lab Guide**

IBM Decision Manger Open Edition

Nigel Crowther – ncrowther@uk.ibm.com

Hands-on Lab

Design Patterns for Big Decisions



NOTICES

This information was developed for products and services offered in the USA.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing

IBM Corporation

North Castle Drive, MD-NC119

Armonk, NY 10504-1785

United States of America

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

**TRADEMARKS**

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at “Copyright and trademark information” at www.ibm.com/legal/copytrade.shtml.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

IT Infrastructure Library is a Registered Trade Mark of AXELOS Limited.

ITIL is a Registered Trade Mark of AXELOS Limited.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Linear Tape-Open, LTO, the LTO Logo, Ultrium, and the Ultrium logo are trademarks of HP, IBM Corp. and Quantum in the U.S. and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

© Copyright International Business Machines Corporation 2020.

This document may not be reproduced in whole or in part without the prior written permission of IBM.

US Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

**Table of Contents**

[1 Introduction 4](#_Toc130064419)

[2 Labs 5](#_Toc130064420)

[2.1 Lab 0 - The Anti-pattern 5](#_Toc130064421)

[Figure 1: Kie Sandbox Start Page 5](#_Toc130064422)

[Figure 2: Habitability Decision Inputs 6](#_Toc130064423)

[Figure 3: Initial Habitability DMN Design 6](#_Toc130064424)

[Figure 4: Logic Type 7](#_Toc130064425)

[Figure 5: Habitability Decision 7](#_Toc130064426)

[Figure 6: Test Result - Habitable 7](#_Toc130064427)

[Figure 7: Test Result - Uninhabitable 8](#_Toc130064428)

[2.2 Lab 1 – Structured Data Pattern 9](#_Toc130064429)

[Figure 8: PlanetInfo Data Type 9](#_Toc130064430)

[Figure 9: Planet DMN Input defined as a PlanetInfo Data Type 10](#_Toc130064431)

[Figure 10: Decision referencing PlanetInfo data type 10](#_Toc130064432)

[2.3 Lab 2 – The Divide and Conquer Pattern 11](#_Toc130064433)

[Figure 11: Pressure Table 11](#_Toc130064434)

[Figure 12: Oxygen Table 11](#_Toc130064435)

[Figure 13: Methane Table 12](#_Toc130064436)

[Figure 14: Atmosphere Table 12](#_Toc130064437)

[Figure 15: Surface Table 13](#_Toc130064438)

[Figure 16: Habitability 13](#_Toc130064439)

[Figure 17: Divide and Conquer Pattern for Habitability 14](#_Toc130064440)

[2.4 Lab 3 – The Tiered Service Pattern 15](#_Toc130064441)

[Figure 18: Tiered Service Pattern 15](#_Toc130064442)

[Figure 19: Atmosphere Decision Service 16](#_Toc130064443)

[Figure 20: Surface Decision Service 16](#_Toc130064444)

[2.5 Lab 5 – Directory Pattern 17](#_Toc130064445)

[2.6 Lab 6 - Generating DMN from Excel 21](#_Toc130064446)

[2.7 Conclusion 22](#_Toc130064447)

# Introduction

In these labs we will implement the DMN patterns describe in the companion document, *Patterns for Scaling Large DMN Projects.*

The labs use **IBM Decision Manager Open Edition**. This lab assumes you have mastered the basics of **DMN** and **KIE Sandbox**.

The Lab will start by building a simple decision service that exhibits a design antipattern. We will improve this design by applying the following patterns:

* The **Structured Data Pattern** to simplify input data in DMN Diagrams
* The **Divide and Conquer Pattern** to split a complex table into smaller tables
* The **Tiered Service Pattern** to split a single DMN file into several smaller DMN Files.

By the end of the lab, you will understand how to build large scale decision projects in DMN.

It is worth noting that not all the patterns may be appropriate for your project. The patterns presented are appropriate for large projects, and for small projects with small decision tables, the patterns may not be required.

The lab material is provided in GIT, all labs except the last can be performed in KIE Sandbox.

**Pre-requisites**

To perform this Lab, you need either:

A remote Virtual Machine from IBM TechZone with a pre-deployed PAM-DM Open Edition tooling to use in this Lab. Please follow instructions in Appendix A to provision and access to an instance of this Virtual Machine.

OR

Your own PC with Chrome or Safari pointing at URL <https://sandbox.kie.org/>

# Labs

## Lab 0 - The Anti-pattern

In this lab, we create an unscalable decision service that decides planet habitability. We will improve the design in subsequent labs.

1. In your Chrome or Safari 16 (or later) browser, open the web site <https://sandbox.kie.org/> :  
     
   Graphical user interface, application, website

   Description automatically generated

Figure 1: Kie Sandbox Start Page

1. Click on the **New Decision** button on the home screen. In the top left corner rename the decision service from Untitled to **Habitability**. Hit the **Enter** button so that the DMN Model Name property is set.
2. Create the following inputs of type **number**: *SurfacePressure, carbonDioxide, methane, oxygen, temperature* and *gravity*:

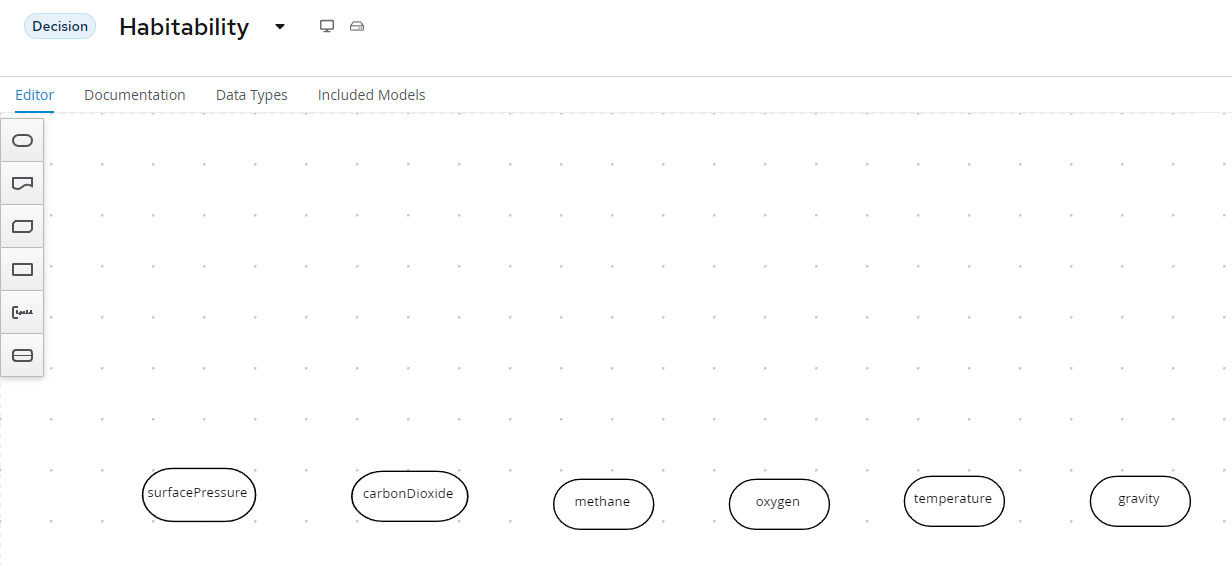


Figure 2: Habitability Decision Inputs

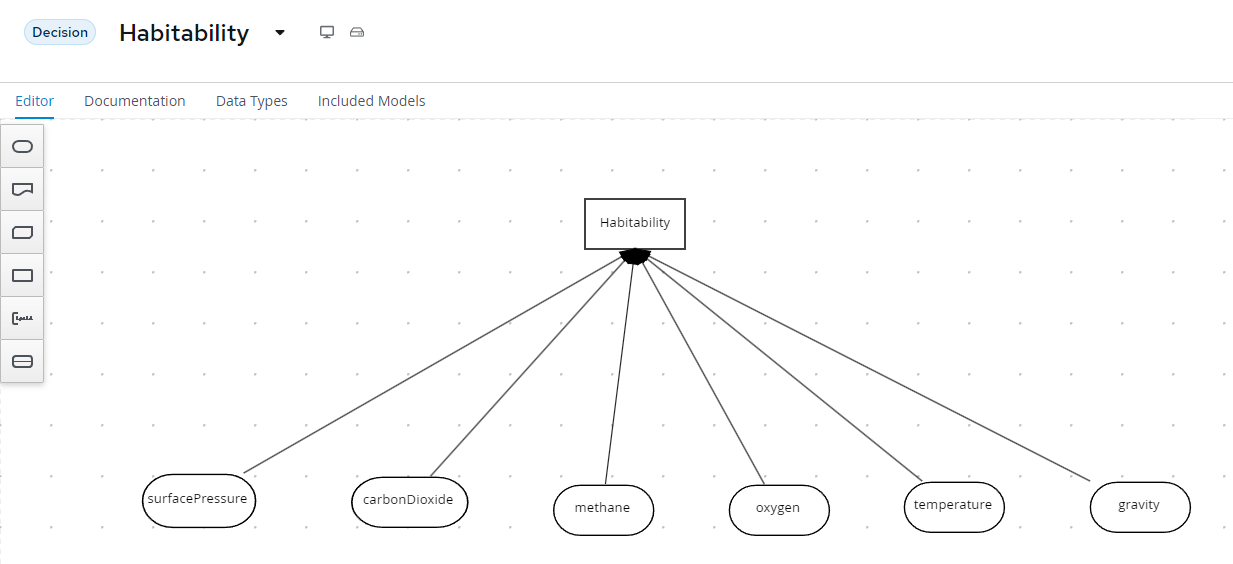
Create Decision *Habitability* and join the inputs to this node:

Figure 3: Initial Habitability DMN Design

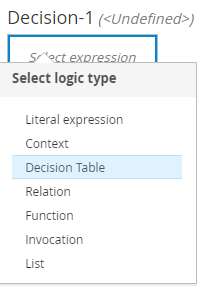
1. Edit **Habitability**. Click on **Select expression** to see a list of available logic types.  
   

Figure 4: Logic Type

1. Select **Decision Table** and enter the following data:

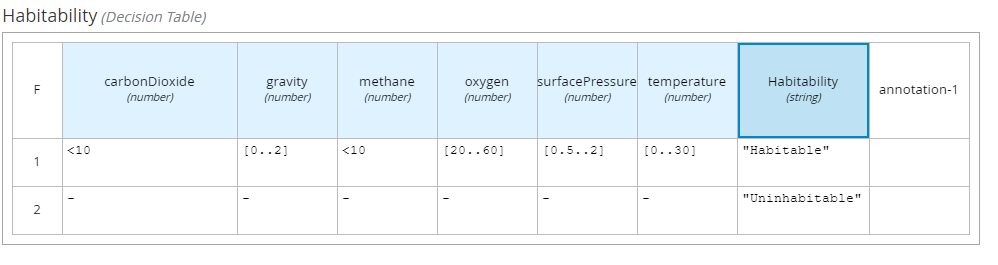


Figure 5: Habitability Decision

1. Test the model with Earth data and the expected result is *Habitable :*

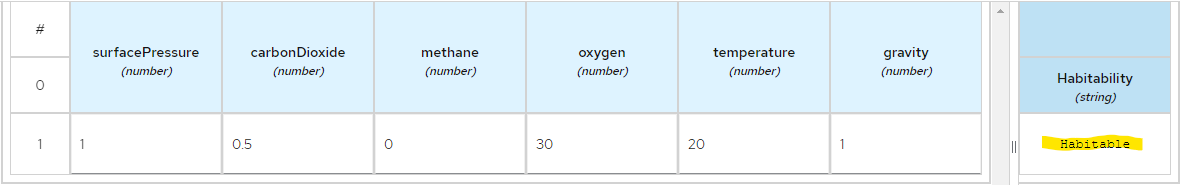


Figure 6: Test Result - Habitable

1. Test the model with Venus data and the expected result is *Uninhabitable :*

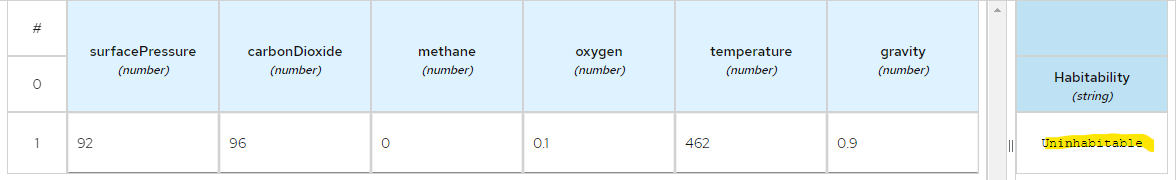


Figure 7: Test Result - Uninhabitable

## Lab 1 – Structured Data Pattern

A poor design of the first iteration is that all decision attributes are specified in the top level DMN. If there are a lot of inputs, its good practice to move them into one or more data types. The DMN model refers to the data types rather than the raw attributes.

This can be modelled in **Kie Sandbox**:

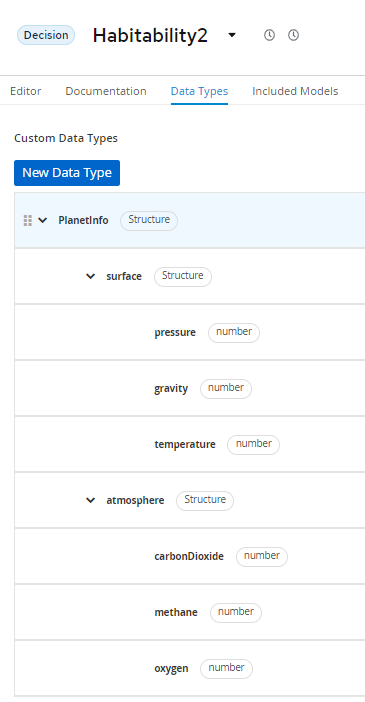


Figure 8: PlanetInfo Data Type

Now we now define just one DMN input called *planet* of type *PlanetInfo*:

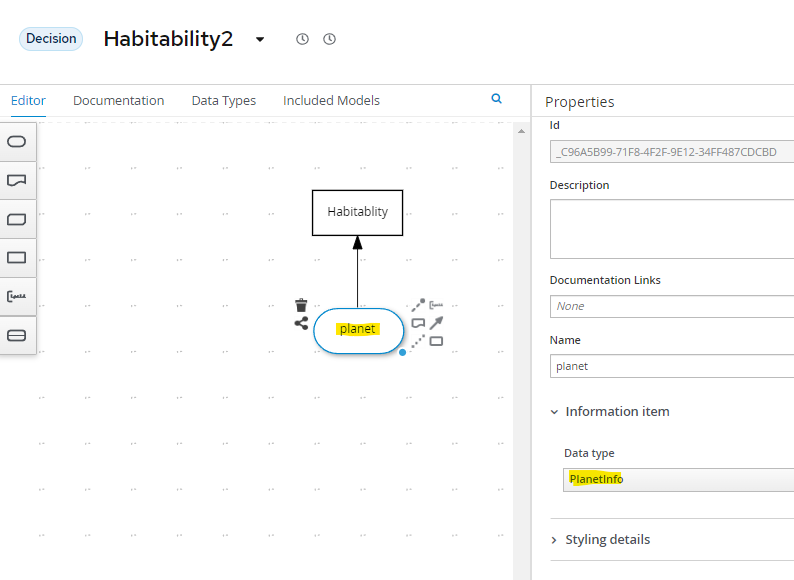


Figure 9: Planet DMN Input defined as a PlanetInfo Data Type

The decision table is the same as the previous lab, but it now references the new data structure:

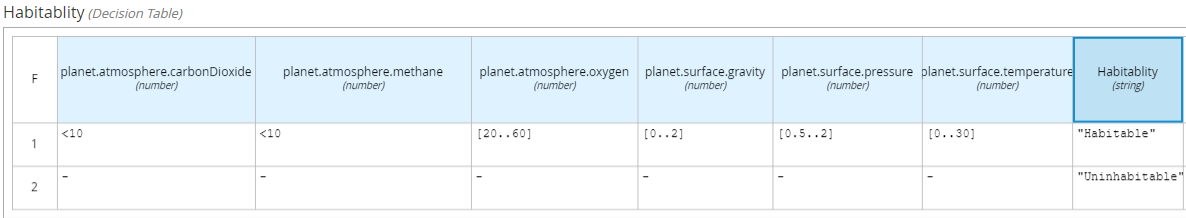


Figure 10: Decision referencing PlanetInfo data type

## Lab 2 – The Divide and Conquer Pattern

To reduce the size of a decision table with many attributes, you can divide and conquer it into smaller parts.

To do this, create a new decision table for each planet attribute. The purpose of these single attribute tables is to reduce the many values of each attribute into a restricted range of enumerated string types. These enumerated types are then applied in aggregated decisions further up in the DMN model. In our example we are reducing all numeric ranges into three enumerated types: *Optimal, Bearable and Deadly.*



Figure 11: Pressure Table

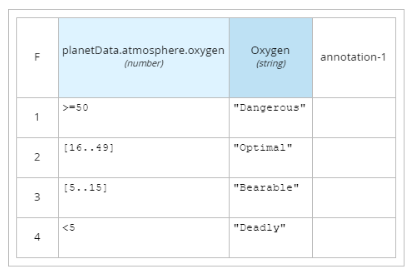


Figure 12: Oxygen Table

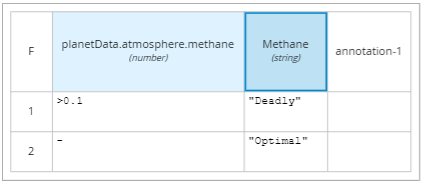


Figure 13: Methane Table

We now combine the decision results of these three tables to define the *atmosphere* decision:

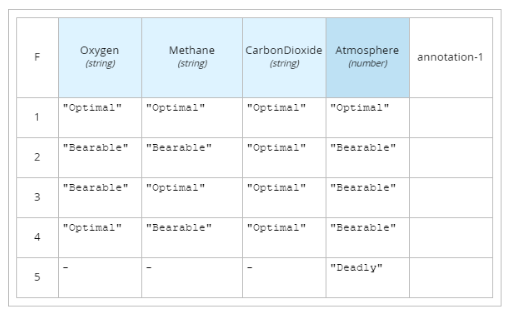


Figure 14: Atmosphere Table

We can apply the same summary pattern to *Pressure, Temperature* and *Gravity* to create the *Surface* Decision:

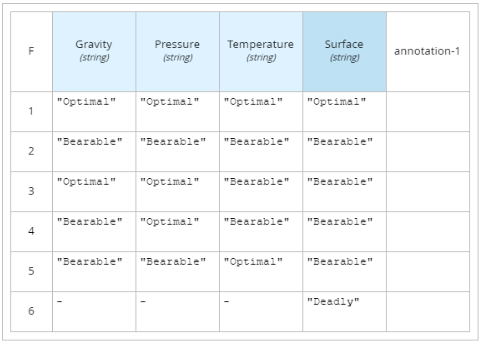


Figure 15: Surface Table

We now join both *Atmosphere* and *Surface* tables to *habitability* which produces the overall decision:

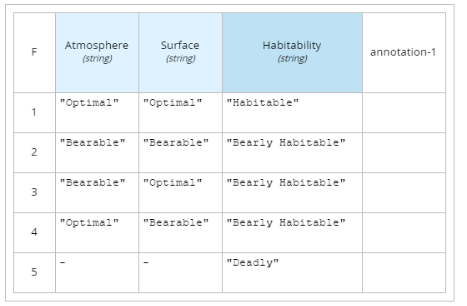


Figure 16: Habitability

All the tables are linked in the DMN diagram as follows:



Figure 17: Divide and Conquer Pattern for Habitability

The **habitability** table has the same behavior as the original DMN design, except now it is easier to maintain. Each planet attribute has its own table making it easier to add attributes and change behavior as there is no dependency on a single table. The last problem we must address is that all this logic is still inside a single DMN file which means the decision tables cannot be edited by separate people and reused in other decisions.

## Lab 3 – The Tiered Service Pattern

Although the DMN defined in the previous section is more maintainable, there is still a design problem. All decision tables are defined within a single DMN file. If additional decision tables and branches were added it would quickly become difficult to maintain. A single DMN file also prevents multiple users from making changes at the same time. To improve on this, we move the second-tier decisions into their own decision service file and then invoke these decisions from the top tier decision service. See figure below, where the second-tier decision services are *AtmosphereDS* and *SurfaceDS* respectively, and the first-tier decision is *Habitability*:

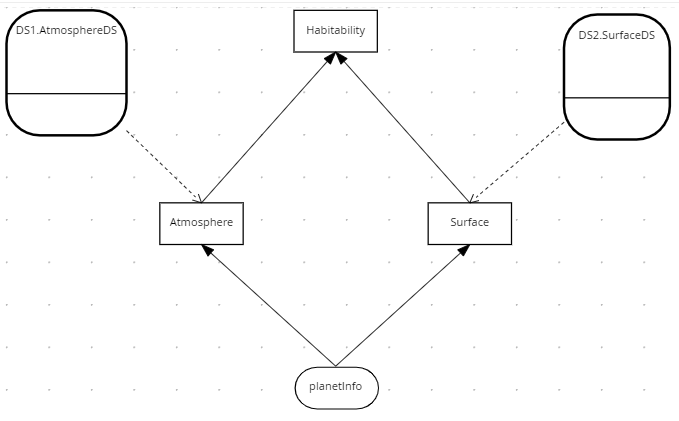


Figure 18: Tiered Service Pattern

The two second tier decision services are *AtmosphereDS* and *SurfaceDS*:

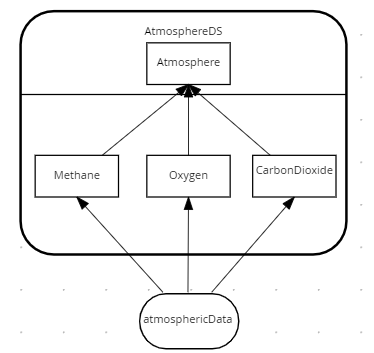


Figure 19: Atmosphere Decision Service

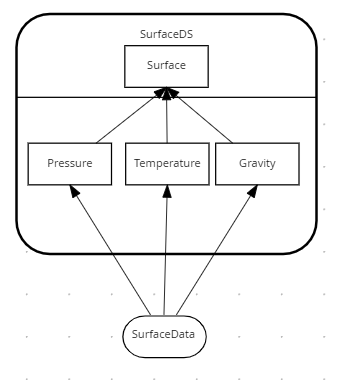


Figure 20: Surface Decision Service

In the Habitability example we have only two tiers – but in practice the pattern could be applied to multiple tiers, with second tier decision services calling third tier services, and so on. It really depends on your project complexity.

## Lab 5 – Directory Pattern

Sometimes the Divide and Conquer Pattern described above does not work. This manifests itself on long thin decision tables with few conditions and many rules. An example of this is SWIFT bank interchange. A simplified model of a SWIFT message is as follows:

**BIC** – A Bank Identifier Code

**Receiving Branch** – The bank branch

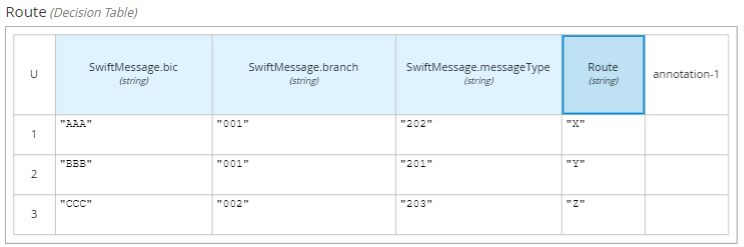
**Message type** – The type of payment

**Route** – The route the message is sent

Routing rules exist to decide where to route a SWIFT message within the secure international banking network. The routes regularly change due to geopolitical and economic decisions, H

Hence the need to routing tables as rules rather than static data.

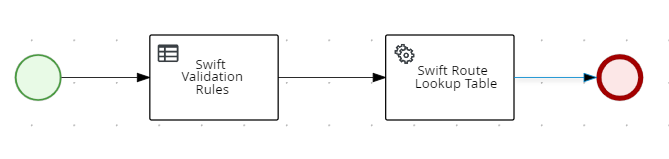
A routing DMN decision table would look like this:



There could be thousands routing rules, based on variates of just these three attributes: *Bic, Branch* and *Message Type*.

So how do we scale this table in a managed way?

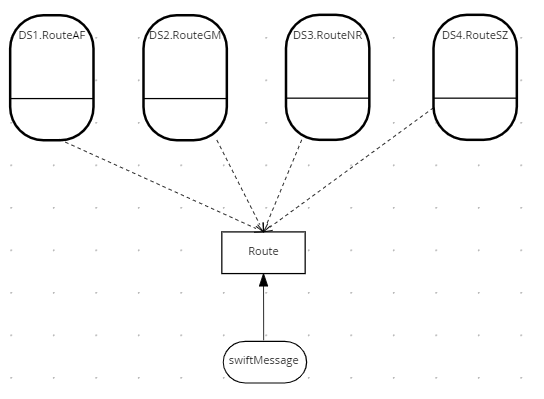
One approach is to step back and consider whether this is really a decision table. Are these ‘rules’ just a reference table? In which case the reference data would be better placed in a database. If rules are required for validation before the database lookup, you could have a hybrid solution using both BPMN and DMN. This solution would look like this:



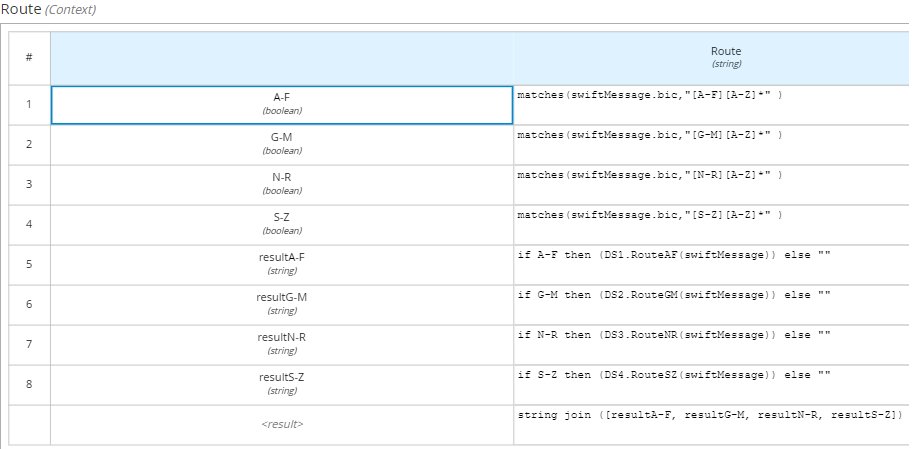
If your rules are definitely rules, maybe because a database query would be too complex or you want to govern changes to the routing, then you can apply the **Directory Pattern**. This pattern splits decision tables the same way an old-fashioned directory alphabetically indexes names and addresses:



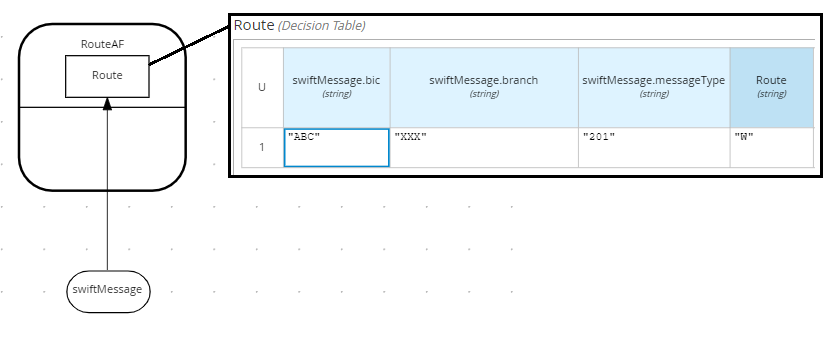
In DMN this is modelled like this:



The key logic is in the *Route* decision. It is complex as it uses regex to determine the correct decision service and then invokes the right one. Note that only the decision service is invoked, not all services. In this case it would be on average four times faster than having a single table with all the rules. Here is the logic:



The decision service for routes A-F is shown below. The other decision services are not shown as the follow the same pattern.



## Lab 6 - Generating DMN from Excel

Many businesses love modelling decisions in Excel. Each worksheet in excel could equate to one or more decision tables. Sometimes the business holds on to their spreadsheet. This can cause maintenance issues as both the spreadsheet and the DMN tables require updating.

There is a solution! The underlying format of DMN is XML and this can be generated. In the following lab we will generate the decision table using a generator that reads Excel data (csv format) and updates the Oxygen table.

## Conclusion

In this lab we presented three design patterns to aid maintainability of large DMN projects. These were:

* The **Structured Data Pattern** to reduce the amount of DMN Input Data
* The **Divide and Conquer Pattern** to divide a single table into smaller tables
* The **Tiered Service Pattern** to divide a single DMN file into several smaller DMN Files.
* The **Service Level Pattern** to reuse decisions and reduce complexity

Consider applying these patterns in your projects and reap the benefits of more maintainable decision logic!