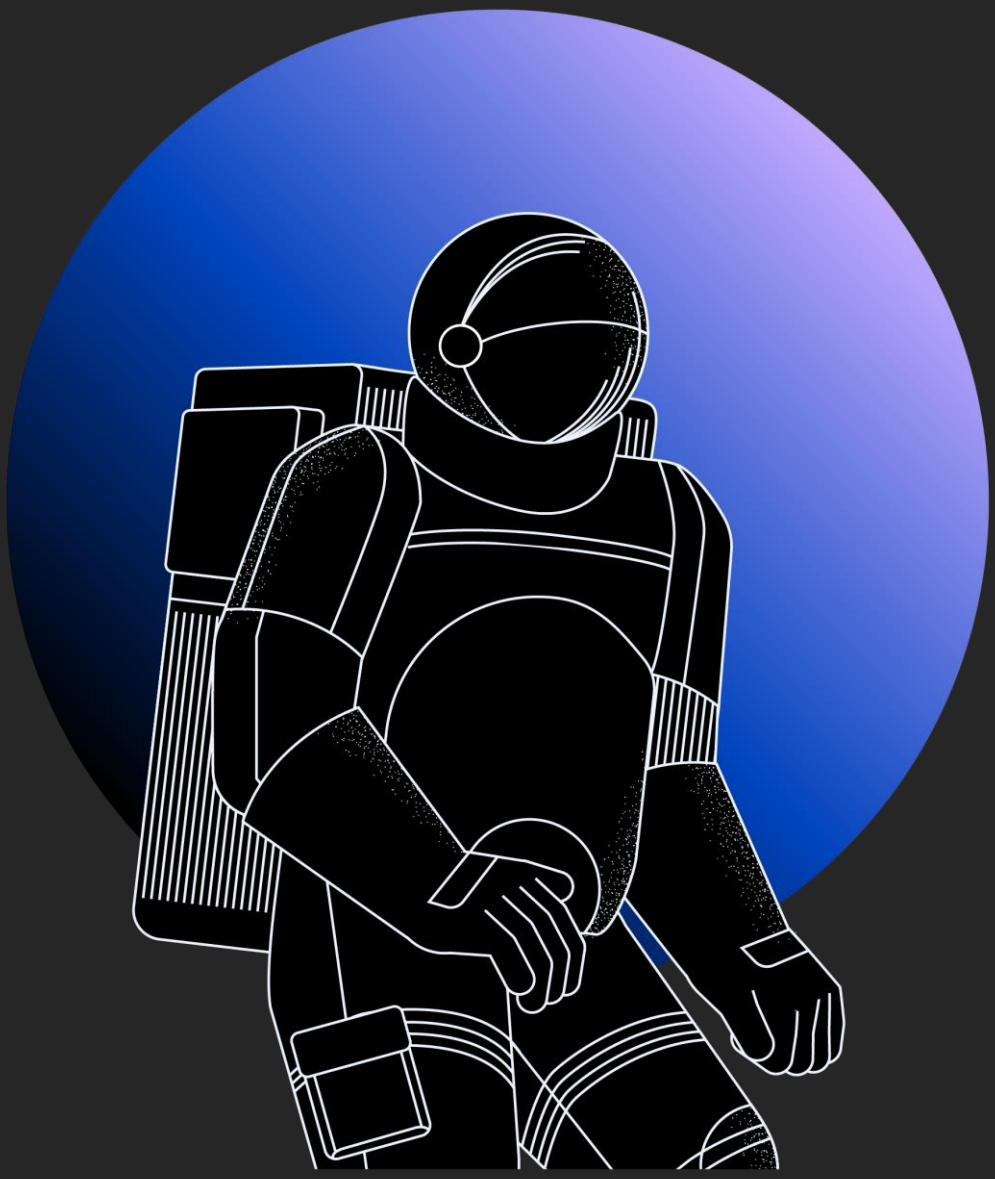
Lab Guide

IBM Decision Manager Open Edition

Nigel Crowther – ncrowther@uk.ibm.com

Hands-on Guide

DMN Beyond the Basics



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 *5*
2 Prerequisites *6*
Lab 1 - Data Types *7*
Introduction *7*
Instructions *8*
Conclusion *13*
Lab 2 – The Divide and Conquer Pattern *14*
Introduction *14*
Instructions *14*
Conclusion *20*
Lab 3 – The Tiered Service Pattern *21*
Introduction *21*
Instructions *23*
Conclusion *25*
Lab 4 – The Index Pattern *26*
Introduction *26*
Instructions *28*
Conclusion *32*
Lab 5 - Hit Policies *33*
Introduction *33*
Instructions *35*
Unique Policy *35*
Any Policy *38*
First Policy *40*
String Collection policy *41*
Numeric Collection policy *43*
Conclusion *44*
Lab 6 - Advanced DMN *45*
Introduction *45*
Instructions *45*
A Quick Tour of the Passenger Priority Service *46*
3 Conclusion *48*

4 Appendix A: Installing KIE Sandbox Extended Services 49

5 Appendix B: Clearing the KIE Sandbox Cache.....50

1 Introduction

In this guide we go beyond basics to build real-world DMN.

The following topics are presented:

- **Data Types**
- **Applying patterns for large projects**
- **Hit Policies**
- **Advanced DMN**

By the end, you will be able to apply the techniques presented in these topics to your projects.

2 Prerequisites

For this lab you need **KIE Sandbox** with *KIE Sandbox Extended Services* running. If you have not already done so, download and install the *KIE Sandbox Extended Services*. See Appendix A.

You will also need a local copy of the following Git repo:

https://github.com/ncrowther/DMNBeyondTheBasics

Click on the link and then click the *Code* button and *Download ZIP*:

Unpack the zip to a local directory and note the location.

For avoiding problems with KIE Sandbox during and between each lab, see Appendix B.

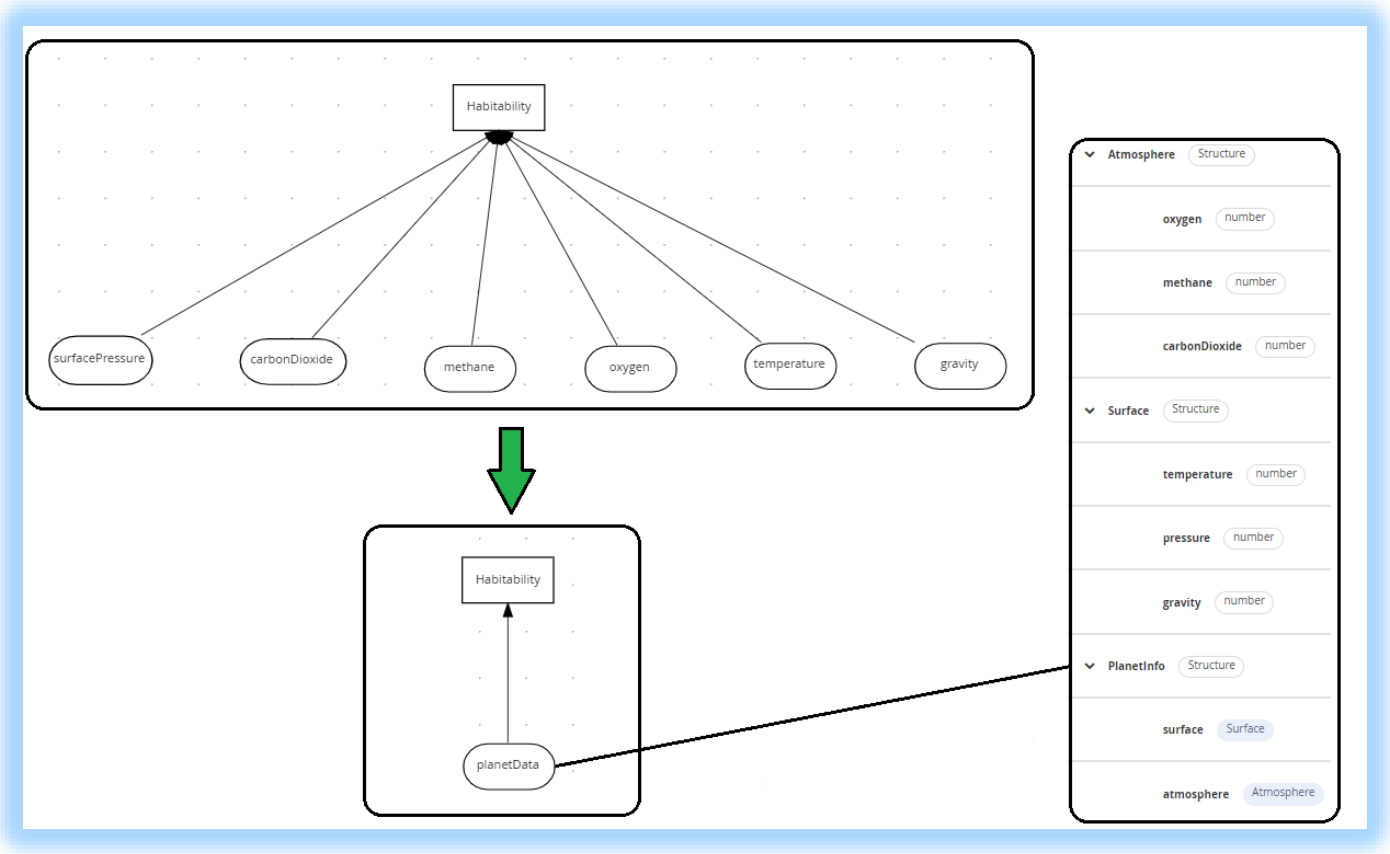
Additional Documentation

Additional documentation on DMN is found here:
https://access.redhat.com/documentation/en-us/red_hat_decision_manager/7.8/html/designing_a_decision_service_using_dmn_models

Lab 1 - Data Types

Introduction

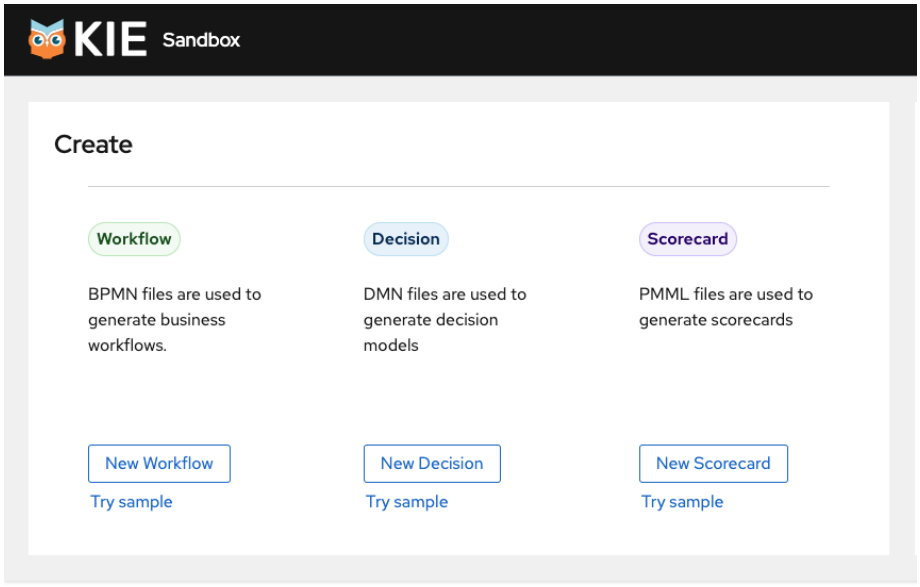
In this lab you will simplify DMN by using data structures. See below:

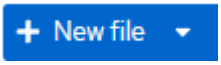


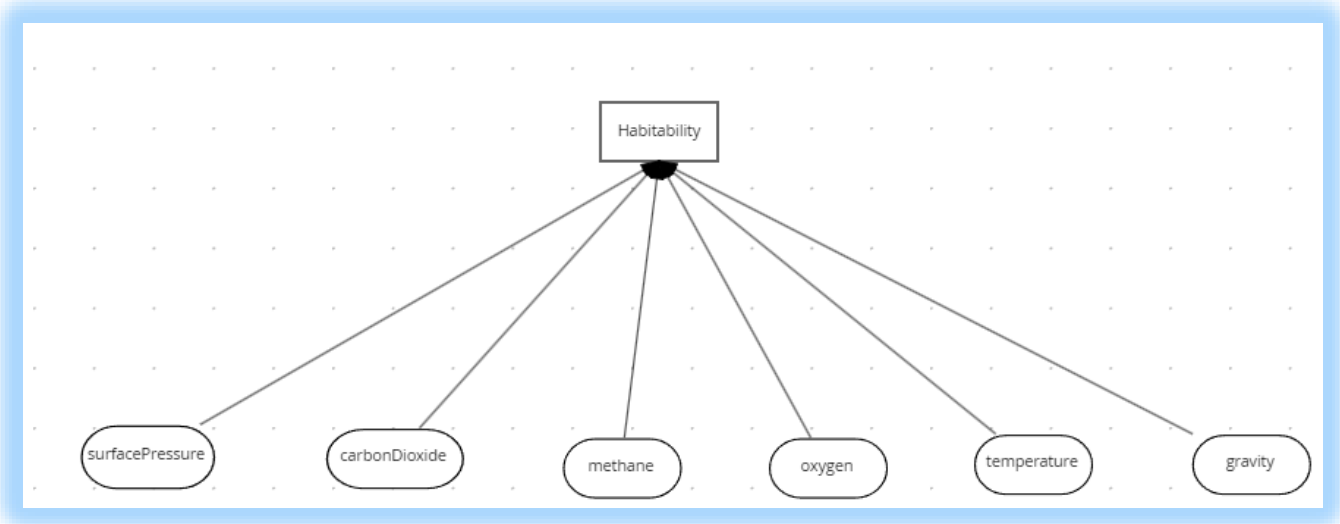
Applying a Data Model to Unstructured Data

Instructions

1. In browsers Chrome or Safari open the web site https://sandbox.kie.org/

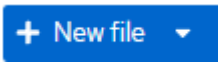


2. Click on **New Decision**.
3. An empty canvas opens[1]. Click *New file*  and then U*pload…*
4. From the downloaded zip contents, Select file: Labs*Lab01**Lab01.dmn*
5. You should see *Lab01*:



[1] If you get a welcome pop-up, you can click on *Skip tour.*

The inputs are simple data types and there are lots of them! This is not the recommended way. If you have many input data, we recommend using data types instead. This we construct next.

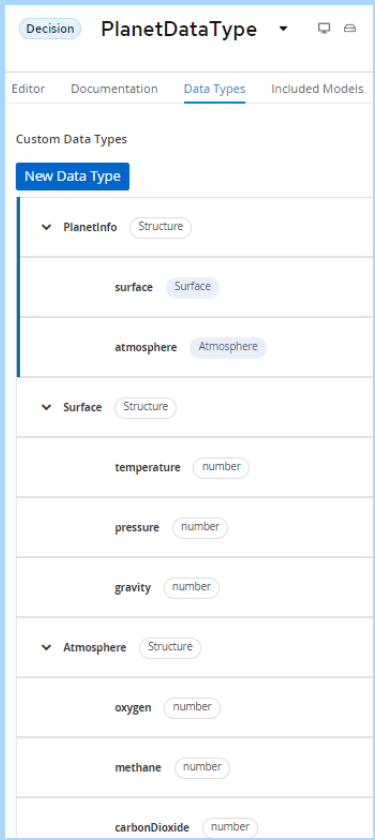
6. Click *New file* 

and then Upload...

7. Select the file: *Lab01/PlanetDataType.dmn.*

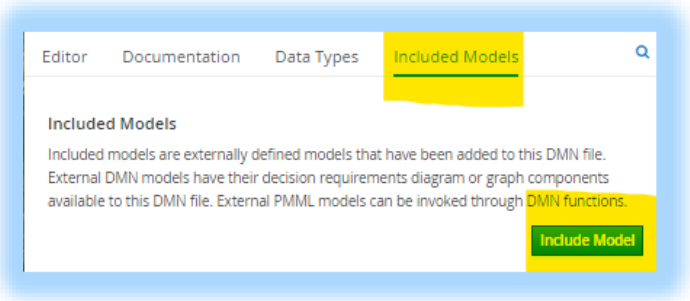
8. Click on the *Data Types* tab:

9. You should see the structure below. This data structure contains all attributes from the original diagram. It contains *surface* and *atmosphere* in a multi-level structure.

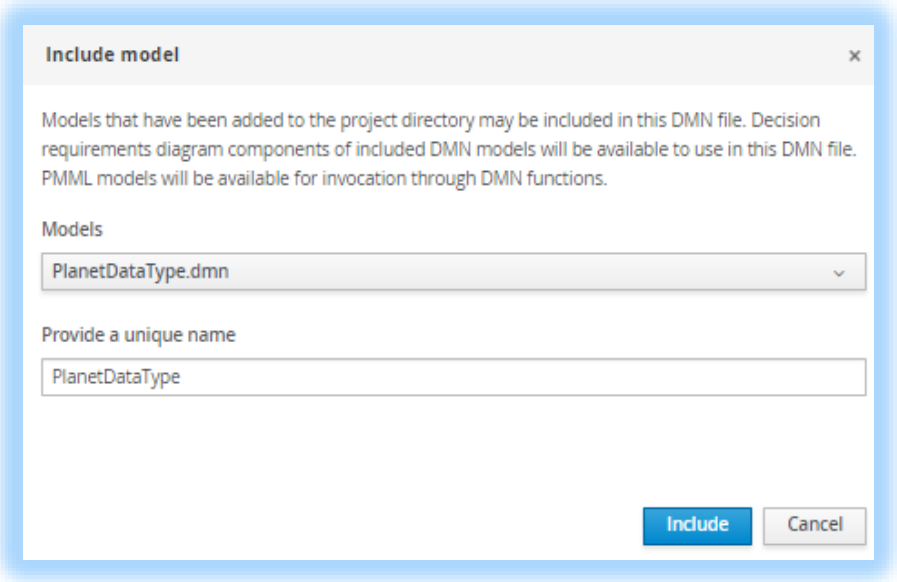


10. Go back to *Lab01* by clicking the drop-down arrow next to *PlanetDataType*:

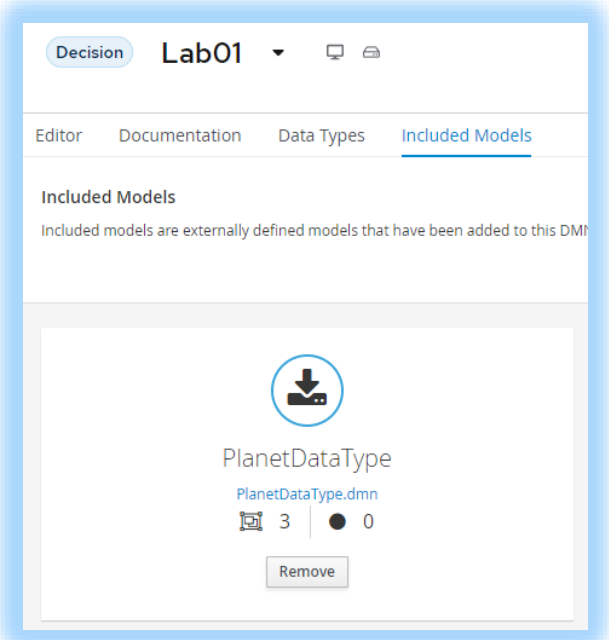
11. Within *Lab01*, Select **Included Models** tab, and then click **Include Model**.



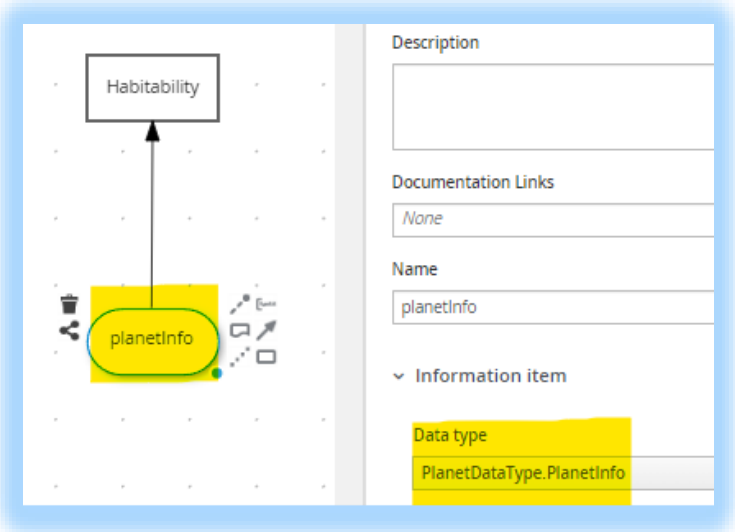
12. Add the *PlanetDataType* model and give it the same name of *PlanetDataType:*



13. Click *Include.* You should see the model has been imported:

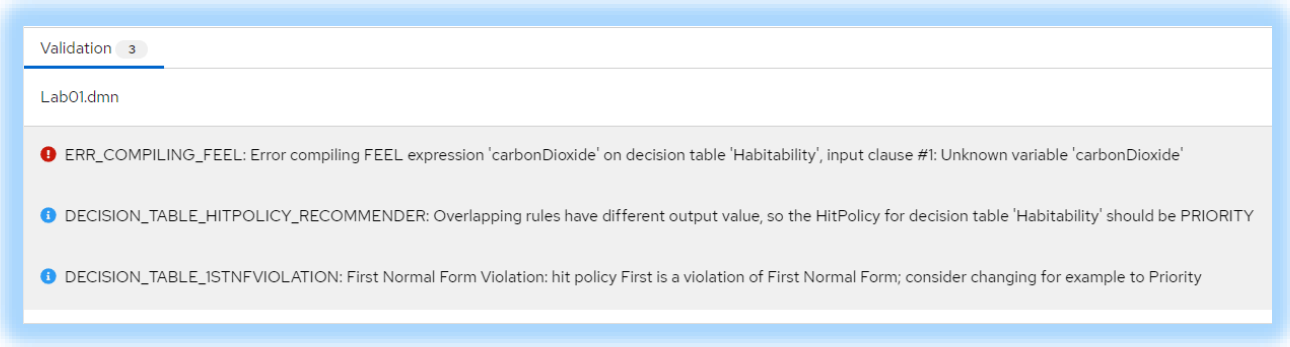


14. Switch to the **Editor** tab.
15. Back in the DMN Diagram, delete all the inputs and replace with one input called *PlanetInfo* assigning it a type of *PlanetDataType.PlanetInfo*.



17. Connect the input to the *Habitability* decision.

16. Click in the Problems button at the bottom right[2]. Then click *Lab01.dmn*. You should see the following errors:

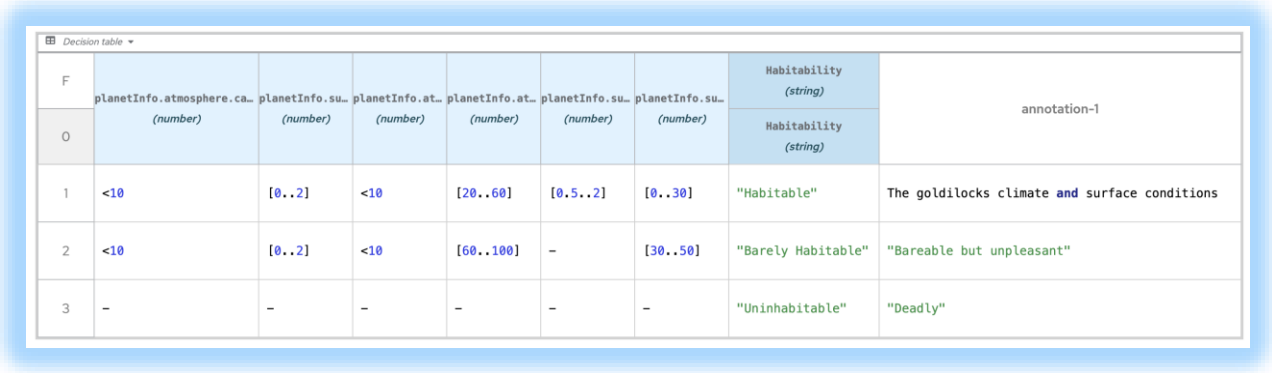


17. The *Habitability* decision table still references the primitive inputs. We need to fix this. Edit the table inputs so that p*lanetInfo* is referenced instead of primitives. The mapping is as follows:

Before Edit	**After Edit**
pressure	planetInfo.surface.pressure
carbonDioxide	planetInfo.atmosphere.carbonDioxide
temperature	planetInfo.surface.temperature
gravity	planetInfo.surface.gravity
methane	planetInfo.atmosphere.methane
oxygen	planetInfo.atmosphere.oxygen

[2] You will need *KIE Sandbox Extended Services* installed. See Appendix A.

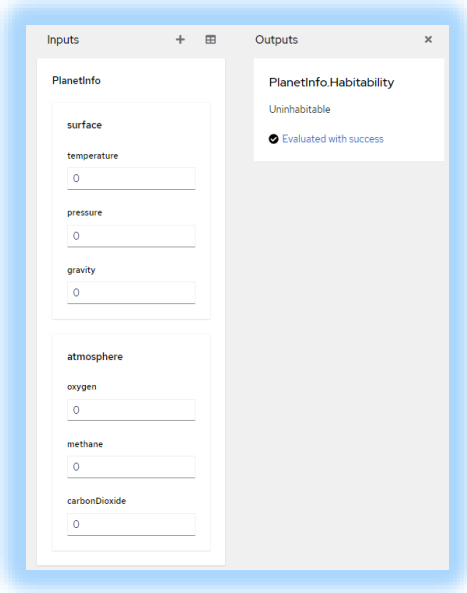
18. Your decision table should look like the one below. You may need to expand the columns to see the full definition:



Decision table

F / 0	planetInfo.atmosphere.ca… (number)	planetInfo.su… (number)	planetInfo.at… (number)	planetInfo.at… (number)	planetInfo.su… (number)	planetInfo.su… (number)	Habitability (string) / Habitability (string)	annotation-1
1	<10	[0..2]	<10	[20..60]	[0.5..2]	[0..30]	"Habitable"	The goldilocks climate and surface conditions
2	<10	[0..2]	<10	[60..100]	-	[30..50]	"Barely Habitable"	"Bareable but unpleasant"
3	-	-	-	-	-	-	"Uninhabitable"	"Deadly"

19. All errors should be clear, and two warnings are left. We will review the warnings in a later lab. Test the model by pressing *Run*. If you have not installed the *KIE Server Extended Services*, now is the time to do so. The runtime panel should show test inputs and outputs:



20. Test with the values below and check the expected results against the actual results[3].

Surface			Atmosphere			Habitability
Temperature	**Pressure**	**Gravity**	**Oxygen**	**Methane**	**CarbonDioxide**	
0	0	0	0	0	0	Uninhabitable
31	2	2	61	9	9	Barely Habitable
25	1	1	20	0	9	Habitable

[3] Can you uncover a defect in the decision table? Hint: When testing "barely habitable", set oxygen to 60. This is a problem of the First hit policy which we will cover in a later lab.

Conclusion

In this lab we refactored a decision with many inputs into a decision with one input associated to a data structure. Combining single input sources into a data structure simplifies DMN.

To continue to the next lab, reset your lab environment as described in Appendix B.

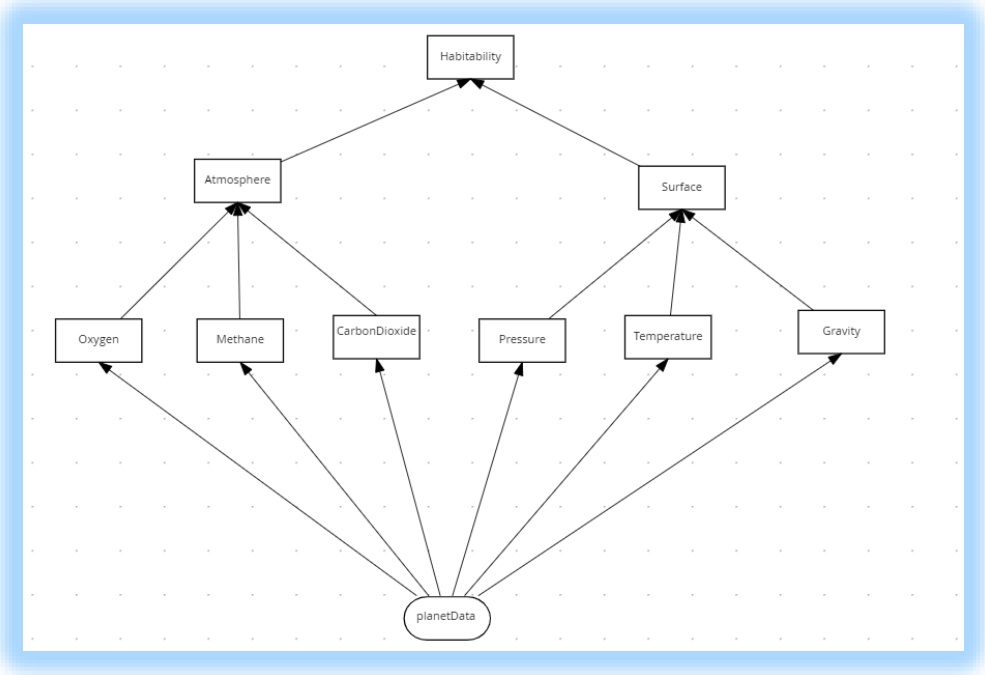
Lab 2 – The Divide and Conquer Pattern

Introduction

The decision table in the previous lab will not scale. If all rule combinations were entered the table would be huge. To reduce the size and complexity we can divide and conquer into smaller parts.

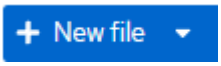
To do this, we create a decision table for each planet attribute and reduce outputs to enumerated values: *Optimal, Bearable* or *Deadly*. Then we apply these outputs in a decision further up in the model. Using this pattern, we split a wide table into several smaller tables.

The tables are linked in the DMN diagram shown below:



The **habitability** table has the same behavior as the original design, with the benefit that it is easier to maintain. Each attribute has its own table making it easier to focus on the decisions for each attribute.

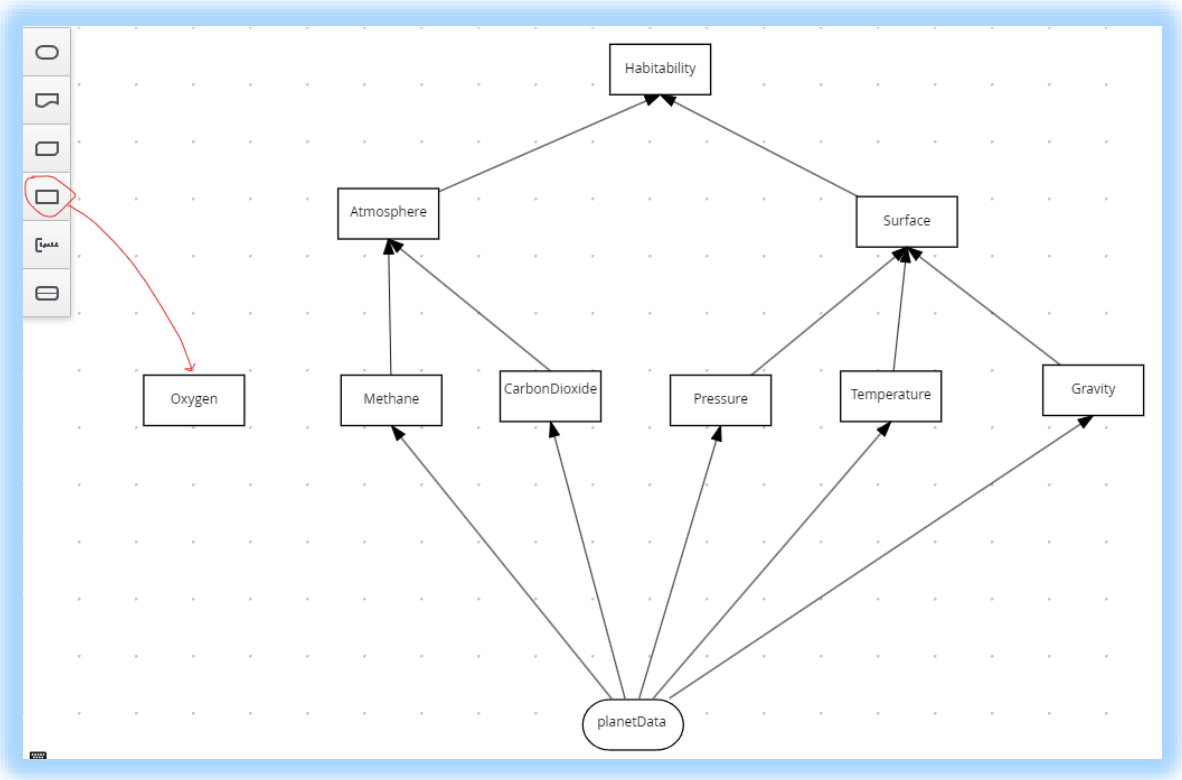
Instructions

1. In a new Chrome or Safari browser, open the web site https://sandbox.kie.org/
2. Click on **New Decision**.
3. An empty canvas opens. Click *New file* 

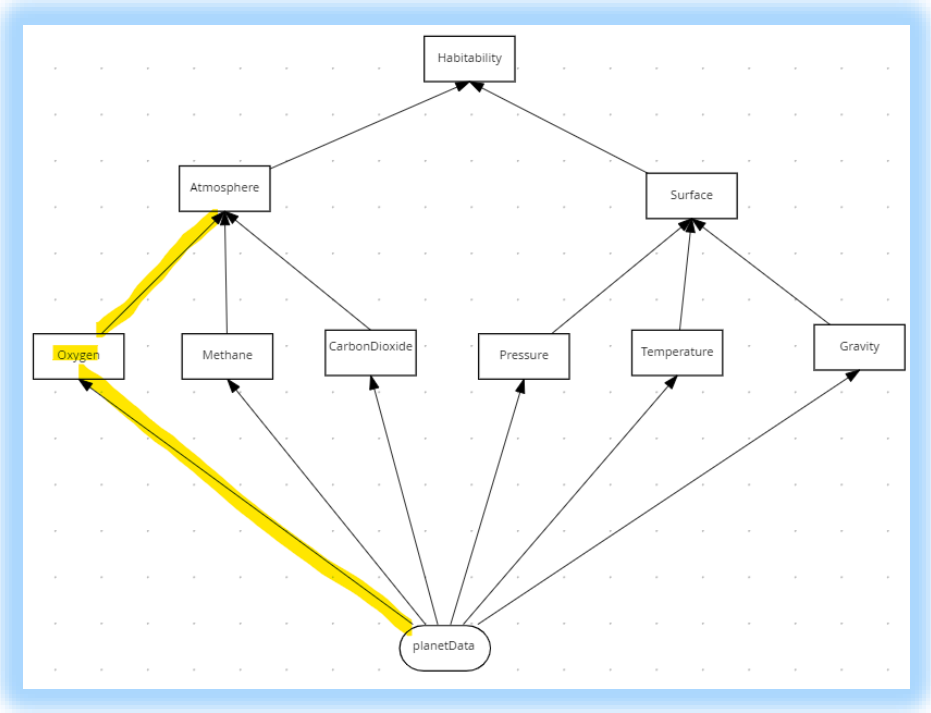
and then U*pload...*

4. Select the file: *lab02/Lab02.dmn*

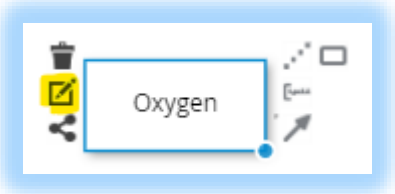
5. The model is nearly complete but is missing the *oxygen* attribute. Create a new Decision node and call it *Oxygen*.



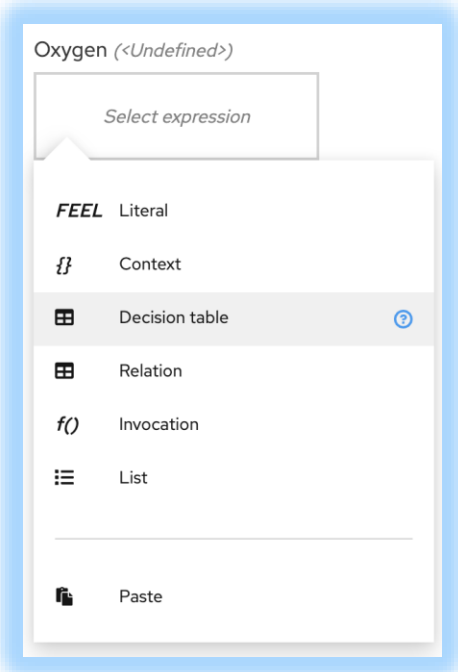
6. Now plug *planetData* to *Oxygen*, and *Oxygen* to Atmosphere. You should have the following:



7. Select *Oxygen* and then click the *edit* button:



8. Select the logic type as *Decision Table*:



9. Create the table below:

Oxygen *(Decision table)*

Decision table

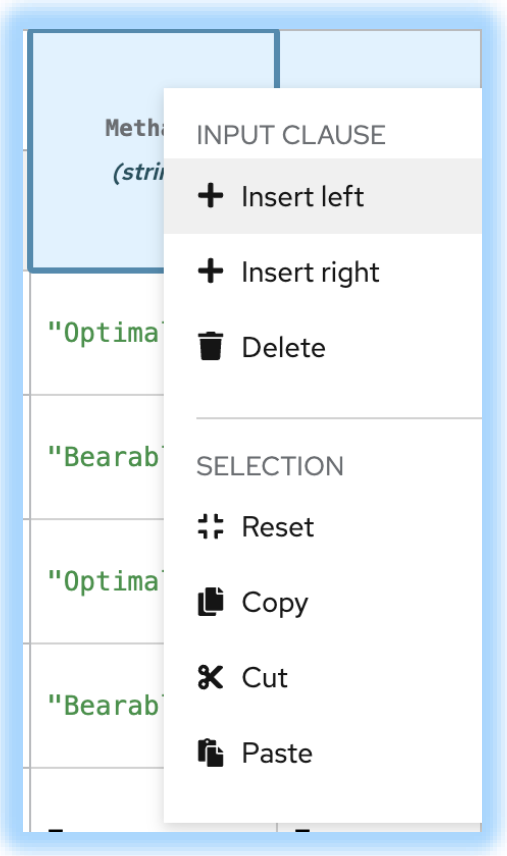
F	planetData.atmosphere.oxygen *(number)*	Oxygen *(string)*
0		Oxygen *(string)*
1	[16..49]	"Optimal"
2	[5..15]	"Bearable"
3	-	"Deadly"

10. Verify the following:

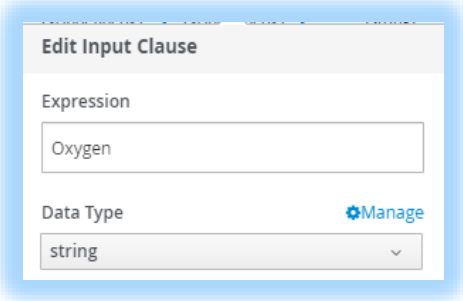
- `planetData.atmosphere.oxygen` is the condition variable. This is case sensitive!
- Numeric ranges `[16..49]` and `[5..15]` are the conditions.
- Enumerated values are in quotes.
- The Hit policy is First (F)
- Oxygen is of type String

11. Edit the *Atmosphere* table to handle the *Oxygen* decision.

12. Right-click on the *Methane* column to add a new column to the left:



13. For the new column, set the Expression as *Oxygen* (case important) and the Data type as *string*.



14. Add four enumerated values (in quotes) as shown below:

Atmosphere *(Decision table)*

Decision table

F	Oxygen *(string)*	Methane *(string)*	CarbonDioxide *(string)*	Atmosphere *(number)*	annotation-1
0				Atmosphere *(number)*	
1	"Optimal"	"Optimal"	"Optimal"	"Optimal"	
2	"Bearable"	"Bearable"	"Optimal"	"Bearable"	
3	"Bearable"	"Optimal"	"Optimal"	"Bearable"	
4	"Optimal"	"Bearable"	"Optimal"	"Bearable"	
5	-	-	-	"Deadly"	

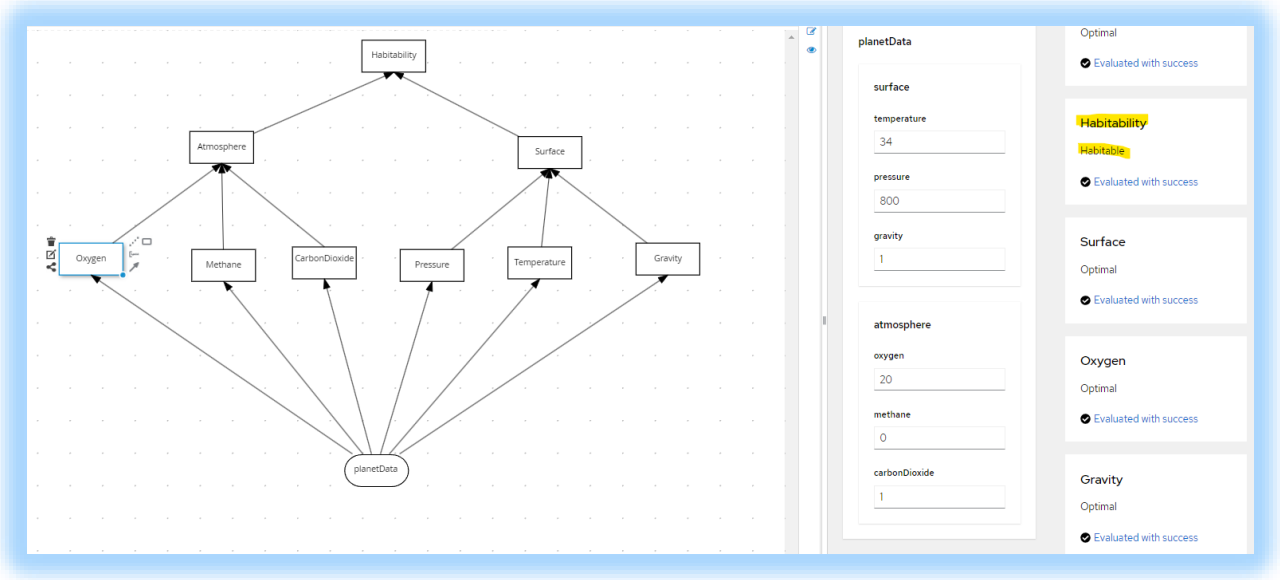
15. Now look at the root decision node, *Habitability*. It combines the decisions in the lower tables to make the final decision on planet habitability:

Habitability *(Decision table)*

Decision table

F	Atmosphere *(string)*	Surface *(string)*	Habitability *(string)*	annotation-1
0			Habitability *(string)*	
1	"Optimal"	"Optimal"	"Habitable"	
2	"Bearable"	"Bearable"	"Bearly Habitable"	
3	"Bearable"	"Optimal"	"Bearly Habitable"	
4	"Optimal"	"Bearable"	"Bearly Habitable"	
5	-	-	"Deadly"	

16. We have now implemented the Divide and Conquer pattern by splitting up one wide decision table into several smaller ones. Let's test the service.
17. Press the *Run* button.
18. Enter values for the inputs until Habitability is *Habitable.* You will need to examine each decision table to find the optimal value for each variable before this decision is reached.



Conclusion

In this lab we refactored the Habitability rules into separate decision services that build up the overall decision. Separating a single decision table helps manage complex decisions with many inputs.

To continue with the next lab, please reset your environment as described in Appendix B.

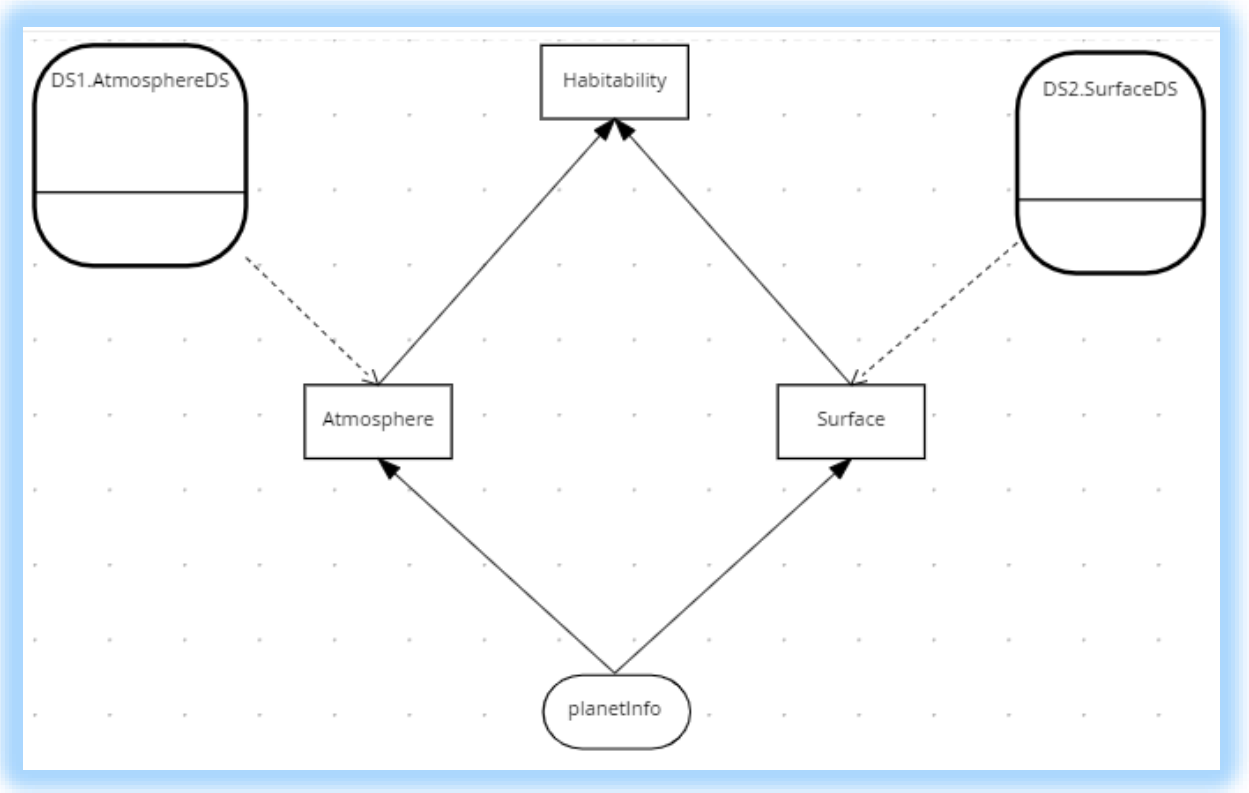
Lab 3 – The Tiered Service Pattern

Introduction

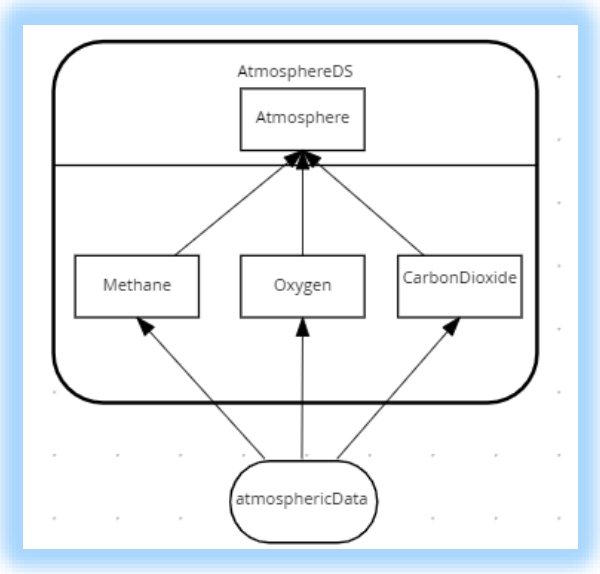
The DMN defined in the previous lab was confined to a single file. This is a problem because:

- The DMN cannot be edited simultaneously by several people.
- The DMN cannot be reused in other decisions.

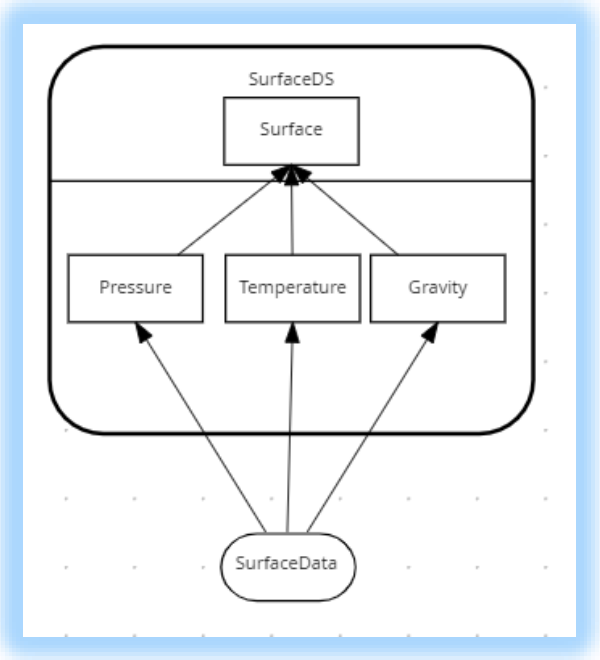
To improve, move the second-tier decisions into decision services and then invoke these decisions from the top tier. See below, where the second-tier decision services are *AtmosphereDS* and *SurfaceDS*:



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



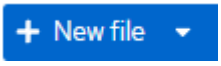
Atmosphere Decision Service



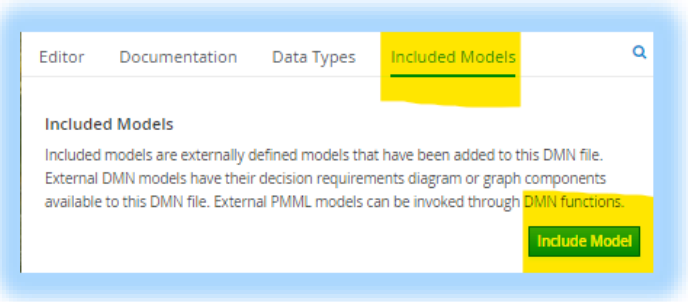
Surface Decision Service

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. The number of tiers depends on project complexity.

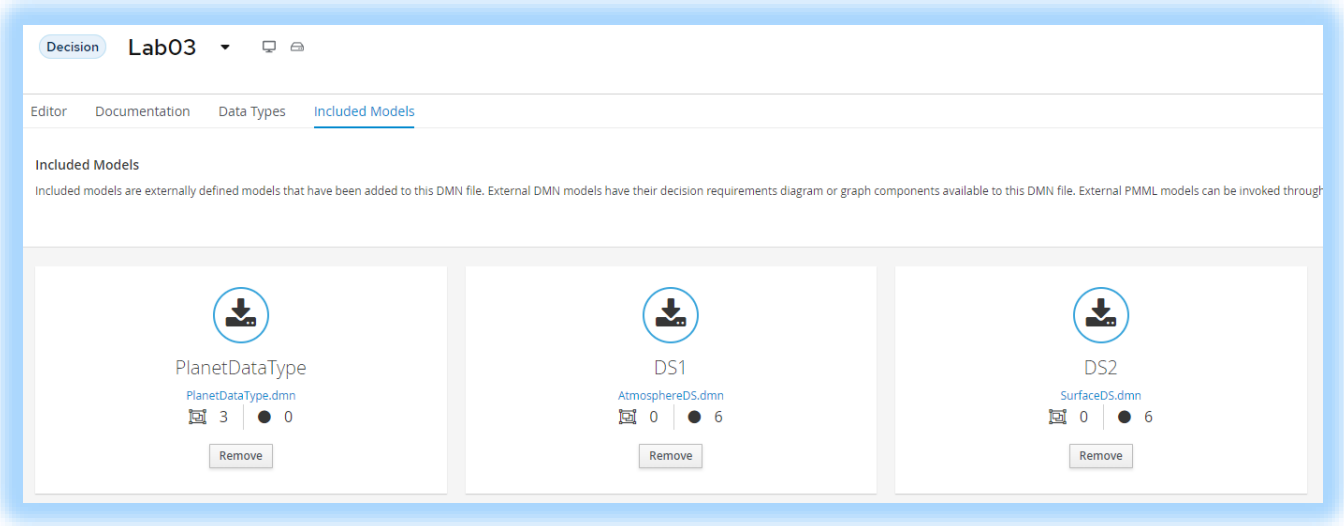
Instructions

1. In a new Chrome or Safari browser open the web site https://sandbox.kie.org/
2. Click on **New Decision**.
3. An empty canvas opens. Click *New file* 

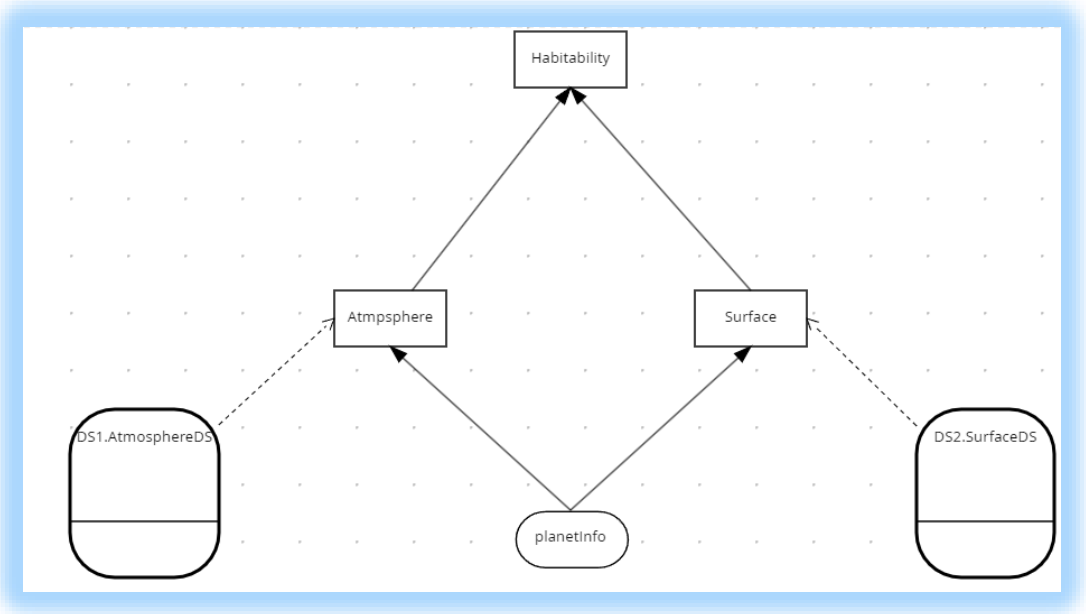
and then U*pload…*
4. Select file: *Lab03/PlanetDataType.dmn*
5. Repeat steps 3 and 4 for the following files:
   Lab03/SurfaceDS.dmn
   Lab03/AtmosphereDS.dmn
   Lab03/Lab03.dmn
6. Within *Lab03*, Select **Included Models** tab.



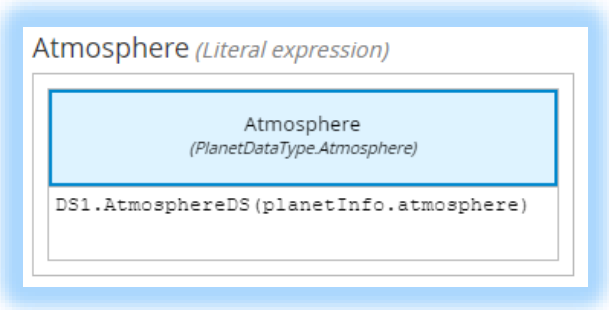
7. You should see the Planet datatype and both decision services are included:



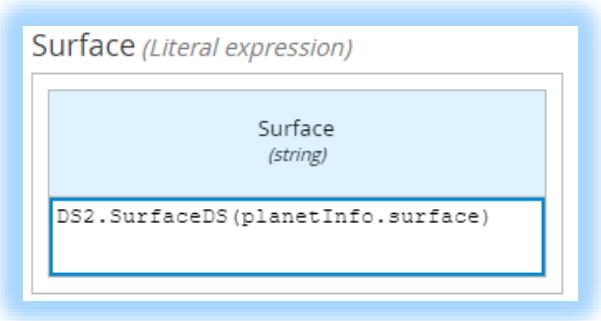
8. Select the *Editor tab.* You should see the following:



9. Edit *Atmosphere* Decision. You should see it calls the *DS1.AtmosphereDS* decision service:



10. Edit *Surface* Decision. You should see it calls the *DS2.SurfaceDS* decision service:



11. Press the *Run* button.

12. Test with the values below and check the expected results against the actual results.

Surface			Atmosphere			Habitability
Temperature	**Pressure**	**Gravity**	**Oxygen**	**Methane**	**CarbonDioxide**	
0	0	0	0	0	0	Uninhabitable
40	1000	2	45	0	5	Fair
25	800	1	33	0	1	Habitable

Conclusion

In this lab we refactored the Habitability rules to call separate decision services. Separating a single decision into multiple services helps scale your DMN projects.

To continue with the next lab, please reset your environment as described in Appendix B.

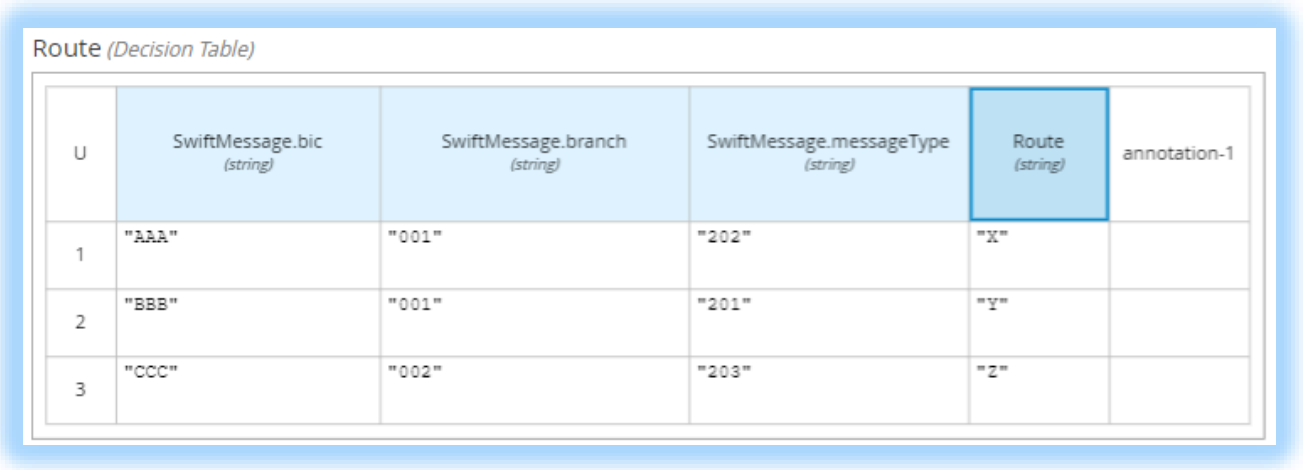
Lab 4 – The Index Pattern

Introduction

The Divide and Conquer Pattern does not work with many rules and few conditions. Use the Index Pattern for this. An example is SWIFT interchange rules. A simplified SWIFT message is:

BIC – A Bank Identifier Code
Receiving Branch – The bank branch
Message type – The type of payment
Route – The route the message is sent

The rules below decide where to route a SWIFT message within the banking network:

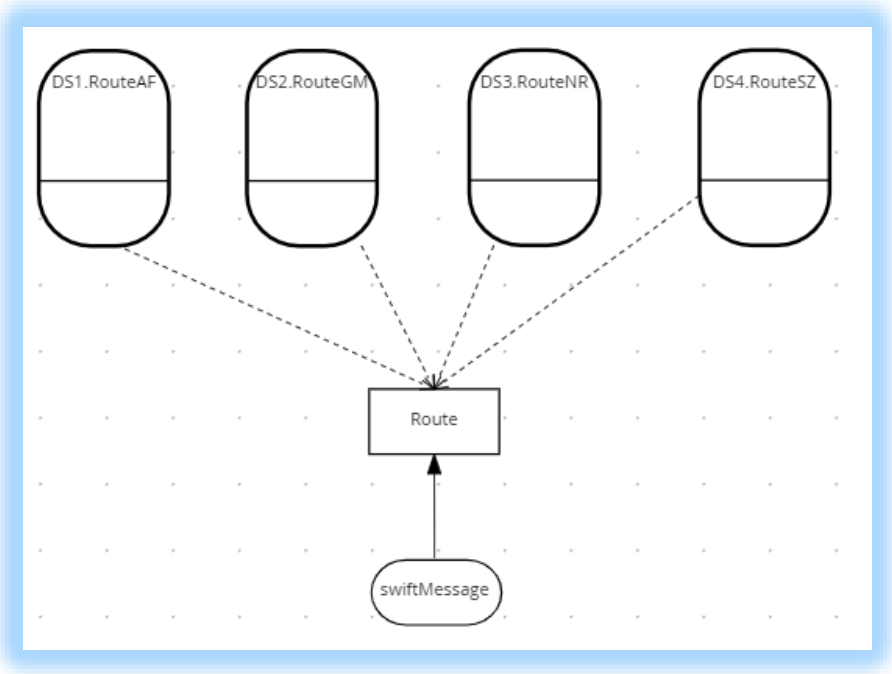


Route *(Decision Table)*

U	SwiftMessage.bic *(string)*	SwiftMessage.branch *(string)*	SwiftMessage.messageType *(string)*	Route *(string)*	annotation-1
1	"AAA"	"001"	"202"	"X"	
2	"BBB"	"001"	"201"	"Y"	
3	"CCC"	"002"	"203"	"Z"	

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

In DMN we could model this using the Index pattern:



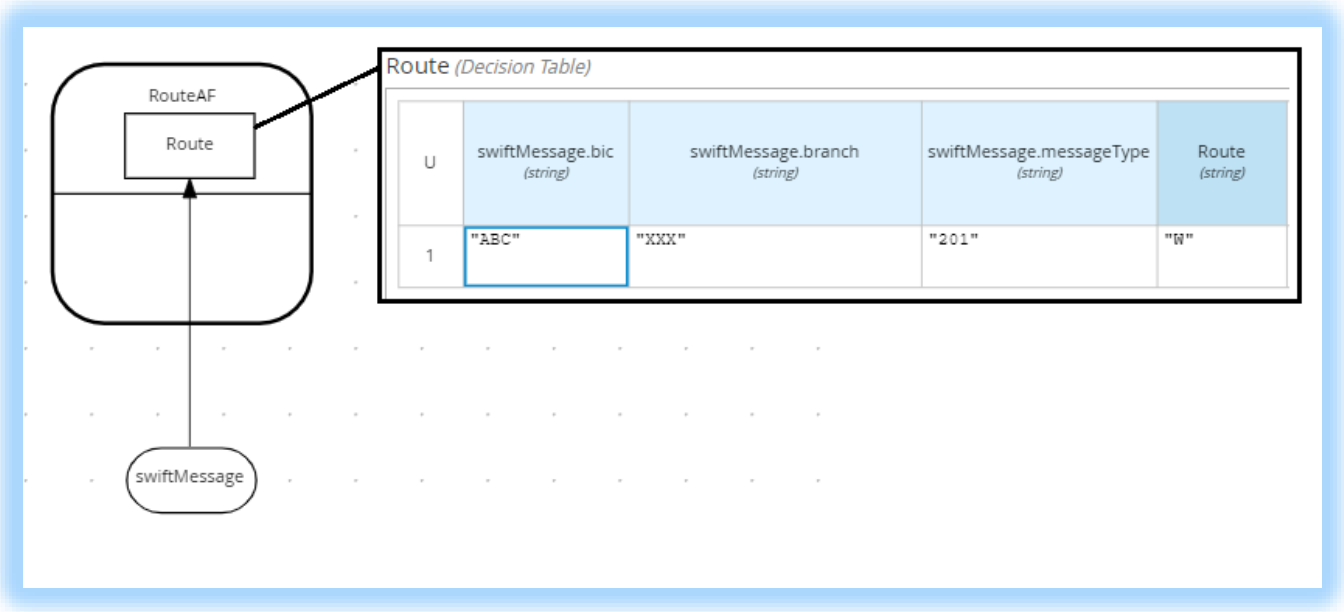
The index logic is in the *Route* decision. It routes the message to an indexed decision service based on the first letter of the BIC. This splits the table into four smaller tables, each alphabetically indexed. There could be further indexing within each of these decision services to split the decision table further.

Here is the logic for the Route decision:

Route *(Context)*

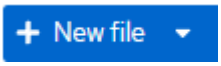
#		Route *(string)*
1	A-F *(boolean)*	`matches(swiftMessage.bic,"[A-F][A-Z]*" )`
2	G-M *(boolean)*	`matches(swiftMessage.bic,"[G-M][A-Z]*" )`
3	N-R *(boolean)*	`matches(swiftMessage.bic,"[N-R][A-Z]*" )`
4	S-Z *(boolean)*	`matches(swiftMessage.bic,"[S-Z][A-Z]*" )`
5	resultA-F *(string)*	`if A-F then (DS1.RouteAF(swiftMessage)) else ""`
6	resultG-M *(string)*	`if G-M then (DS2.RouteGM(swiftMessage)) else ""`
7	resultN-R *(string)*	`if N-R then (DS3.RouteNR(swiftMessage)) else ""`
8	resultS-Z *(string)*	`if S-Z then (DS4.RouteSZ(swiftMessage)) else ""`
	<result>	`string join ([resultA-F, resultG-M, resultN-R, resultS-Z])`

The indexed decision service handles each part of its allocated alphabet range. The service handling BICs starting A-F is shown below.



The decision services handling BICs G-M, N-R and S-Z follow the same pattern.

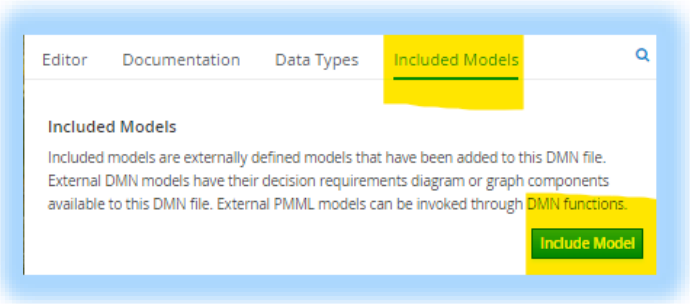
Instructions

1. In browsers Chrome or Safari open the web site https://sandbox.kie.org/
2.
3. Click on **New Decision**.
4. An empty canvas opens. Click *New file* 

and then U*pload...*
5. Perform a multiple select on the Lab04 files:

   Lab04/SwiftDataStructure.dmn
   Lab04/RouteAF.dmn
   Lab04/RouteGM.dmn
   Lab04/RouteNR.dmn
   Lab04/RouteSZ.dmn
   Lab04/SwiftRoutingRulesStart.dmn

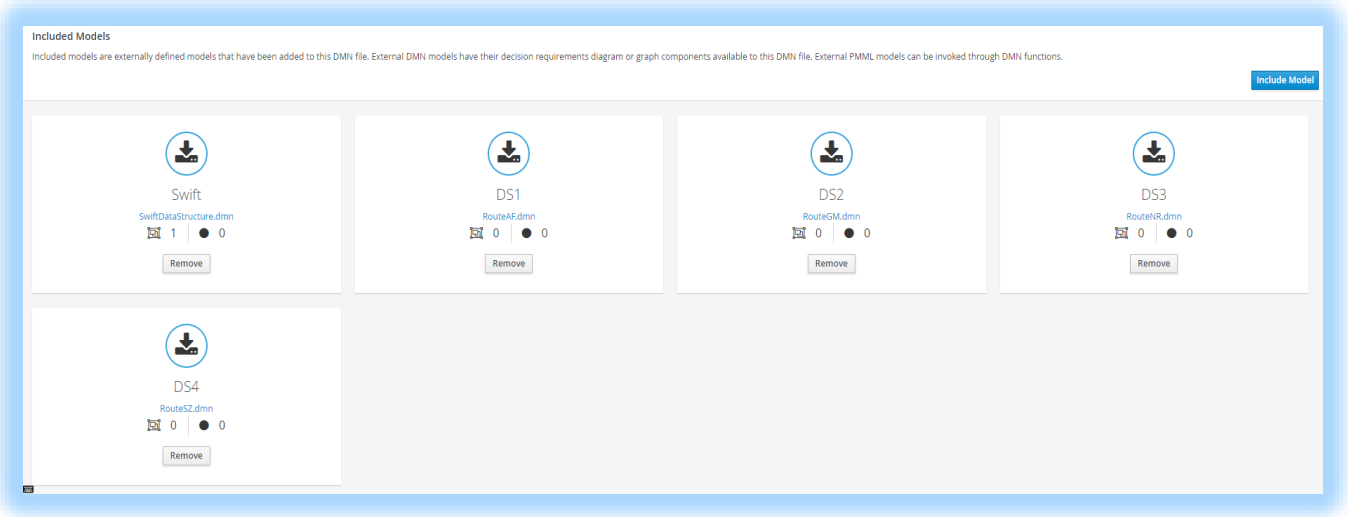
6. Within *SwiftRoutingRulesStart*, select **Included Models** tab, and then click **Include Model**



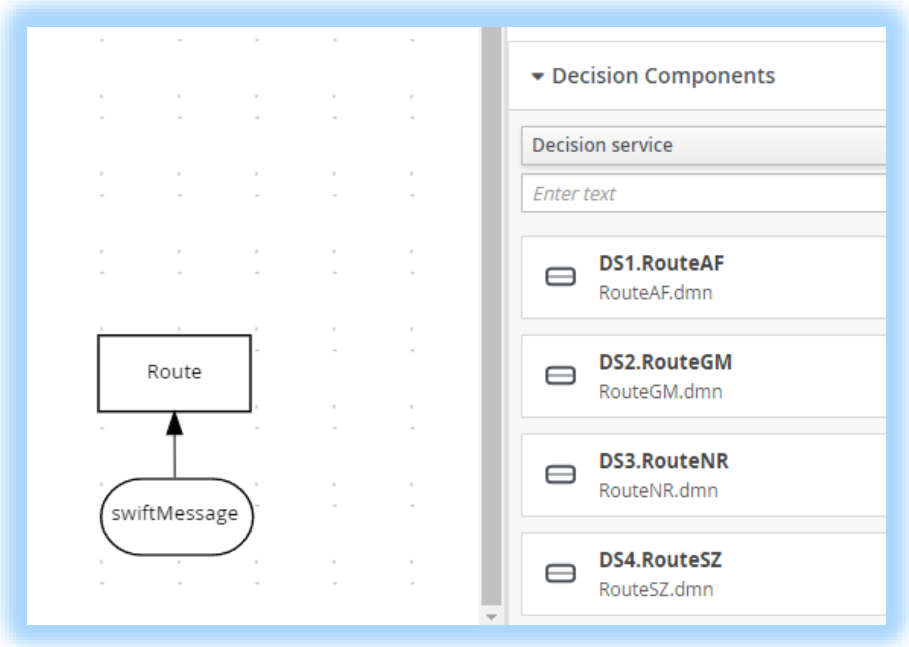
7. Include the following DMN Files as models:

DMN File	Name
SwiftDataStructure.dmn	**Swift**
RouteAF.dmn	**DS1**
RouteGM.dmn	**DS2**
RouteNR.dmn	**DS3**
RouteSZ.dmn	**DS4**

8. Once you have done this, you should see the following models included:

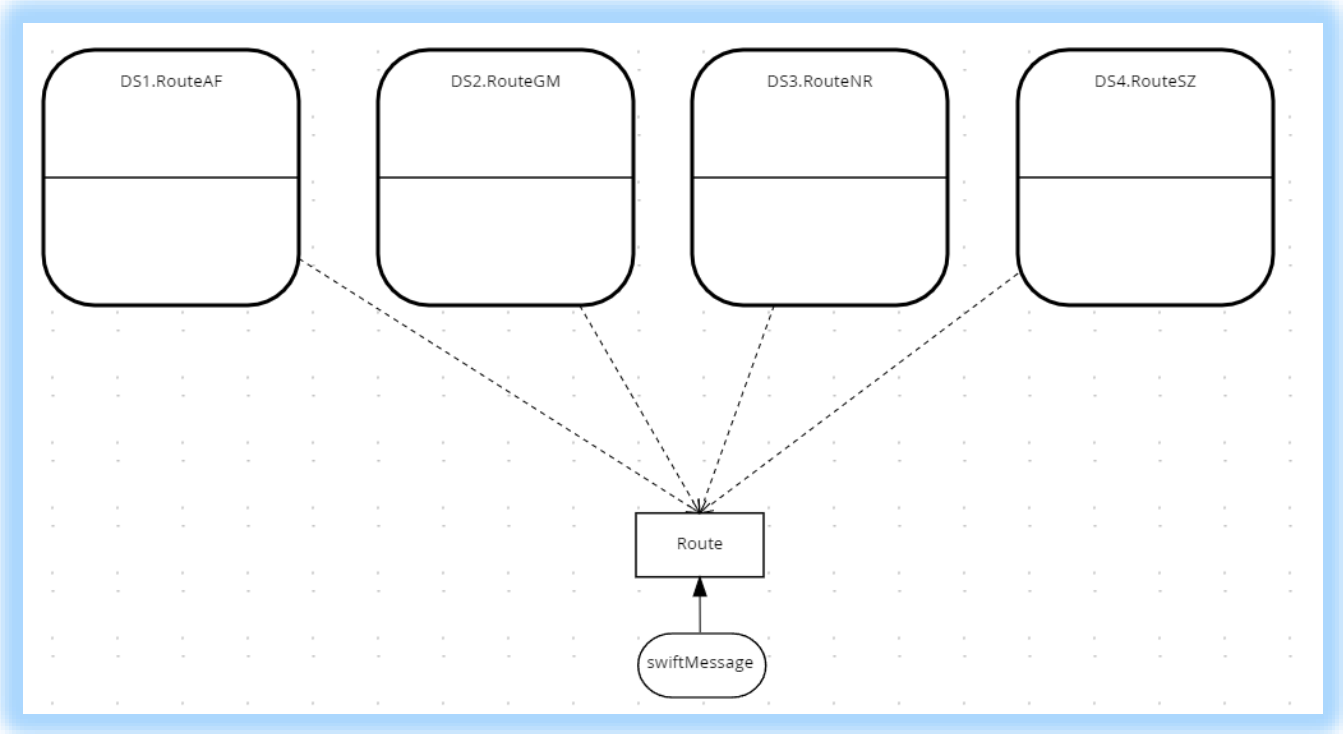


9. Select the *Editor tab.*
10. Select the Decision Navigator button on the far right .
11. Filter by Decision Service. You should see the following:

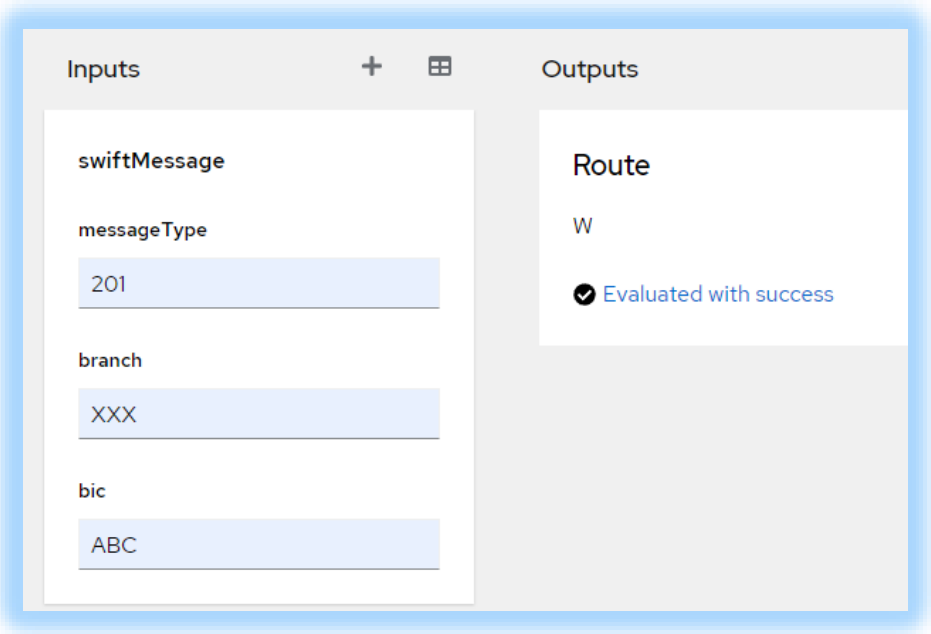


12. Drag and drop all four decision services to the diagram.

13. Using the arrow connector, connect *the decision services to Route*. You should see the following:

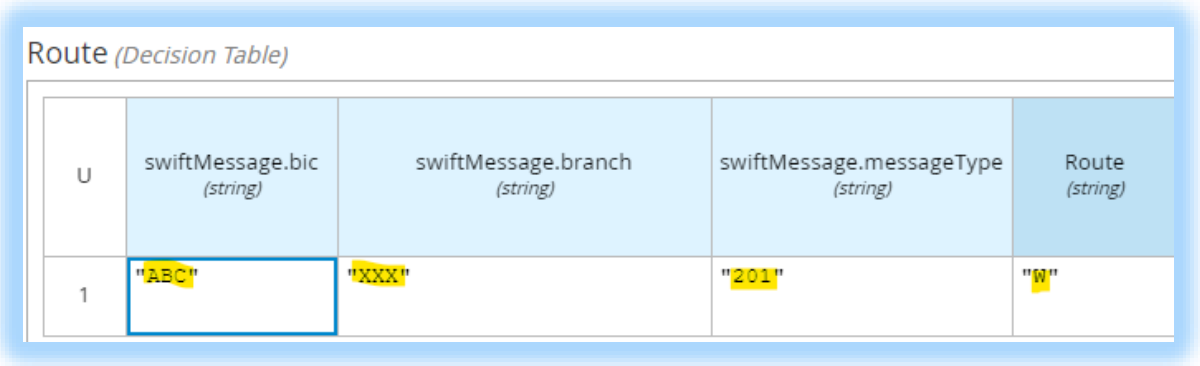


14. Test the decision service by pressing *Run*.
15. Enter the following values:



The output should be W. The Route decision determined the BIC started with 'A' and routed it to RouteAF. This decision service determines that a Swift message of ABC/XXX/201 is routed to W.

16. Verify this by selecting the RouteAF decision service and viewing the decision table.



17. Now test the following Swift messages:

Swift Message	Route
GBC/XXX/201	**X**
NBC/XXX/201	**Y**
SBC/XXX/201	**Z**

You should see that each message is routed to a different decision service. If there are execution errors, talk to your instructor.

18. Now add a new message: **ZBC/XXX/201** with route **Z1**. Where would you put this rule?

Conclusion

In this lab we split a tall narrow decision table into more manageable chunks using the Index Pattern.

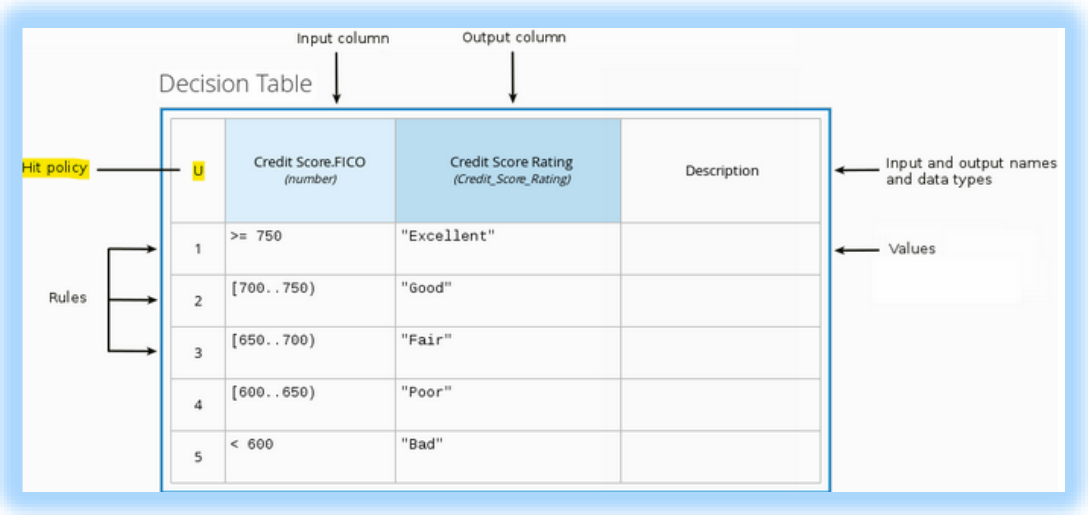
To continue with the next lab, please reset your environment as described in Appendix B.

Lab 5 - Hit Policies

Introduction

Decision tables do not always behave the same way. Some tables count outcomes, some are go / no go decisions and others require precise reasoning. Different hit policies lead to different results and require different ways of thinking about the decision table.

In DMN the hit policy is specified in the top left of the decision table. See below:



It is recommended to select the appropriate hit policy *before* adding rows to your decision table. The common policies are:

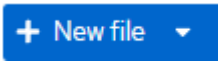
Hit Policy	Description	When to use
Unique (U)	Permits only one rule to match. Any overlap raises an error.	For detailed reasoning. Ensures your rules cover all cases and are complete
Any (A)	Permits only one rule to match. But allows overlaps.	As above but less strict enforcement of overlaps
First (F)	Rules are evaluated from top to bottom. Rules may overlap, but only the first match counts.	For concise decision tables where a simple go / no go decision is needed rather than complete reasoning.
Collect (C)	Aggregates values in an arbitrary list.	For multiple row matches. **String** aggregator: • <None> - matching instances returned as list • Count – Matching instances counted **Number** aggregator: • SUM – matching instances added • Count – matching instances counted • Min – minimum value • Max – maximum value

Additional hit policies not covered in this lab are:

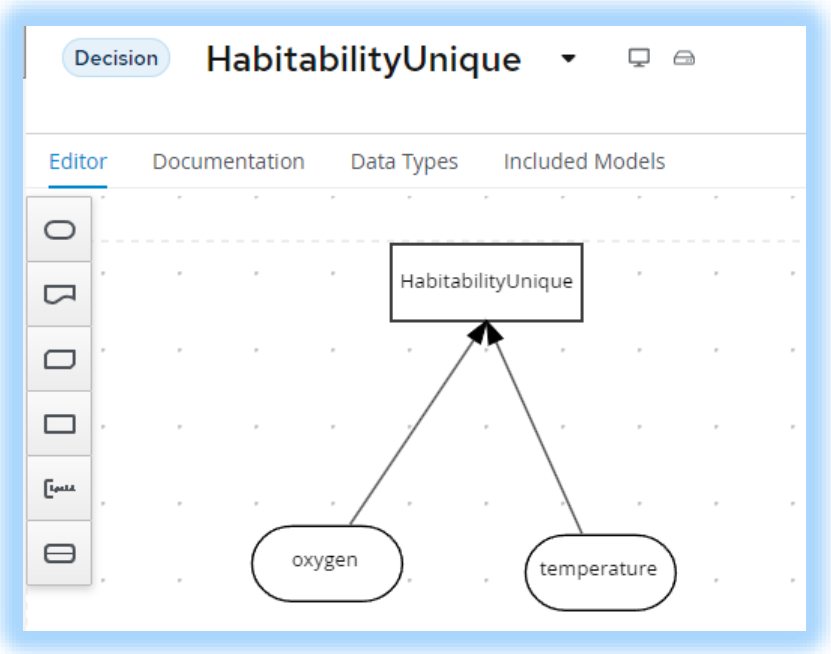
- **Priority** which behaves like **Any**
- **Rule Order** and **Output Order** which behave like **Collect.**

Instructions

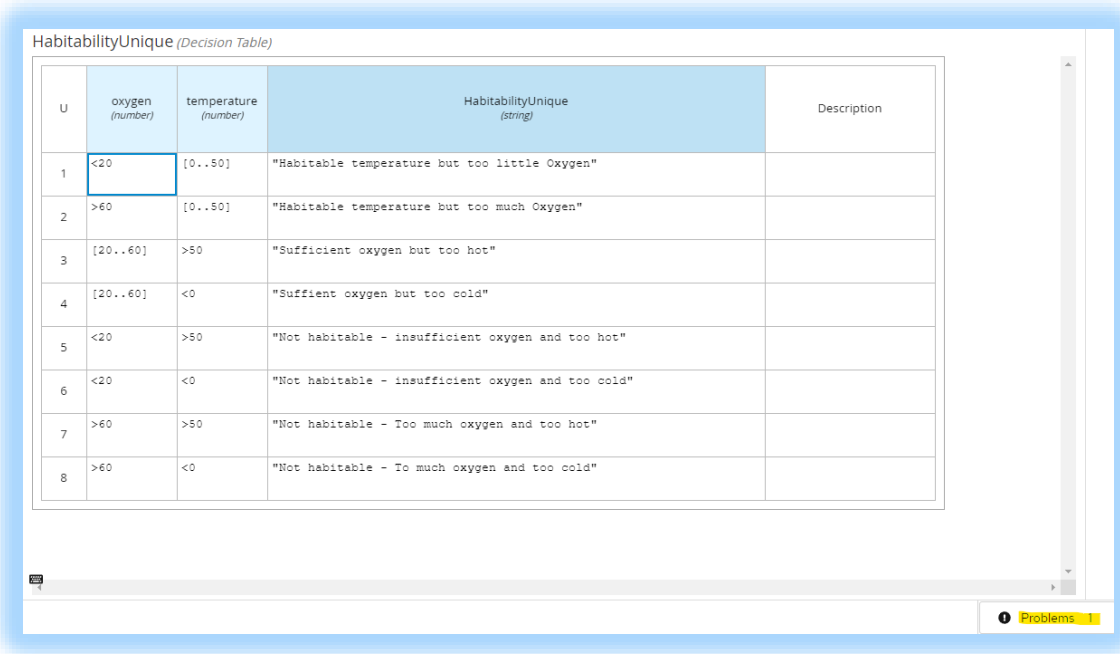
Unique Policy

1. In a new Chrome or Safari browser open the web site https://sandbox.kie.org/
2. Click on **New Decision**.
3. An empty canvas opens. Click *New file* 

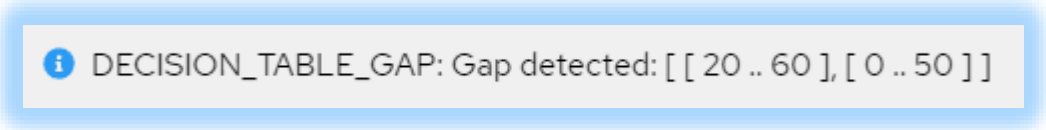
and then U*pload...*
4. Select the file: *Lab05/HabitabilityUnique.dmn*. You should see this:



5. Edit the *HabitabilityUnique* decision by clicking it and selecting Edit .
6. Delete the first row by right clicking row 1 and select *Delete*.
7. After deleting the first row you should see this:

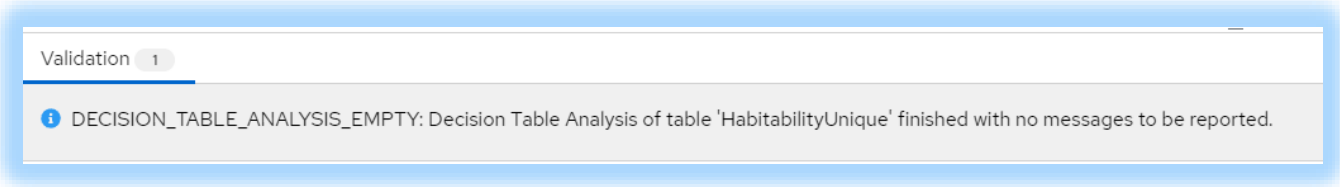


8. Click in the ❗ *Problems* button in the bottom right. There is a gap warning:



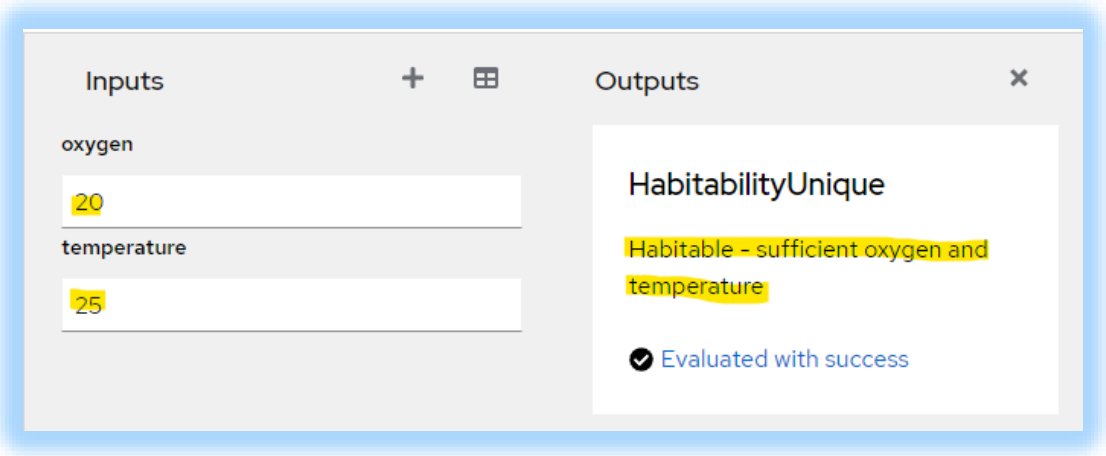
9. The gap created from the deleted row has been detected.

10. Fix this error by hitting **Ctrl-Z** to undo the change[4]. The deleted row should reappear, and the analysis warning should disappear. You should see:



11. Test by pressing R*un*. Enter Oxygen *20* and Temperature *25*:

[4] If this does not work, start form step 1 .



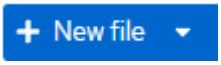
12. The decision correctly evaluates to "*Habitable - sufficient oxygen and temperature*".

Conclusion

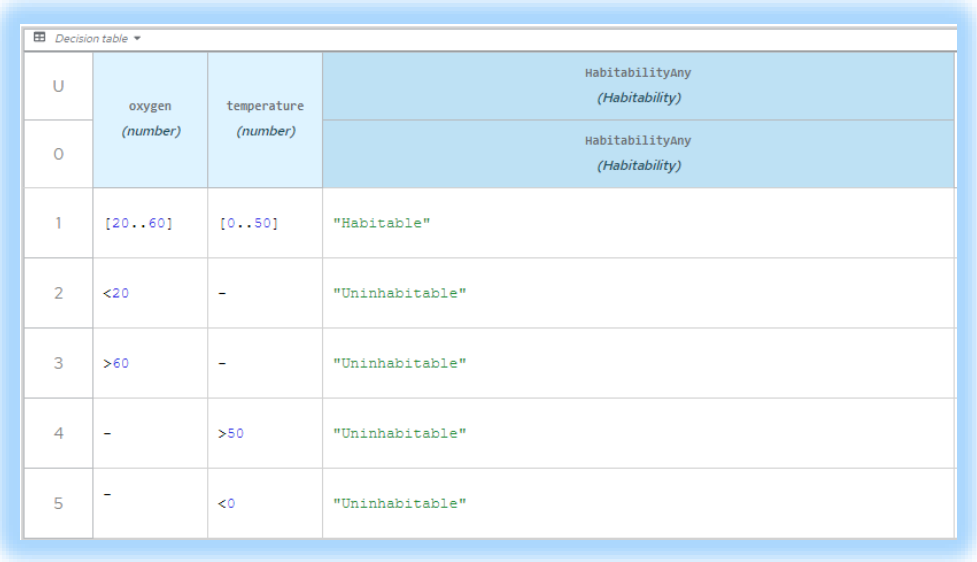
The *Unique* policy reasons over every possible input. This is useful for decisions requiring traceability. For example, an applicant my want to know the decision behind their rejected mortgage application. A second advantage of the Unique policy is that you can order rows in any order to get the same result.

The disadvantage of the Unique policy is that it requires the table to cover every possible condition which can be too strict for simple decisions.

Any Policy

1. In a new Chrome or Safari browser open the web site https://sandbox.kie.org/
2. Click on **New Decision**.
3. An empty canvas opens. Click *New file*  and then U*pload…*

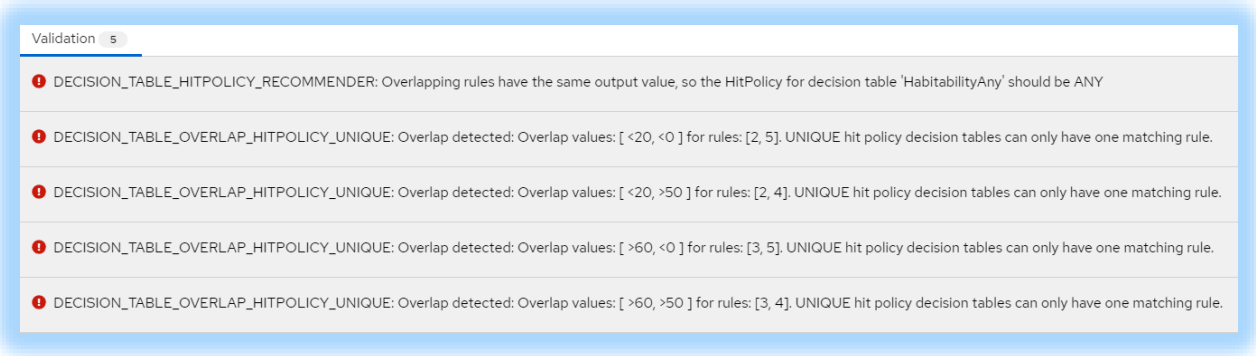
4. Select the file: *Lab05/HabitabilityAny.dmn*
5. Edit the *HabitabilityAny* decision by clicking it and selecting Edit .
6. You should see this:



Decision table

U	oxygen *(number)*	temperature *(number)*	HabitabilityAny *(Habitability)*
0			HabitabilityAny *(Habitability)*
1	[20..60]	[0..50]	"Habitable"
2	<20	-	"Uninhabitable"
3	>60	-	"Uninhabitable"
4	-	>50	"Uninhabitable"
5	-	<0	"Uninhabitable"

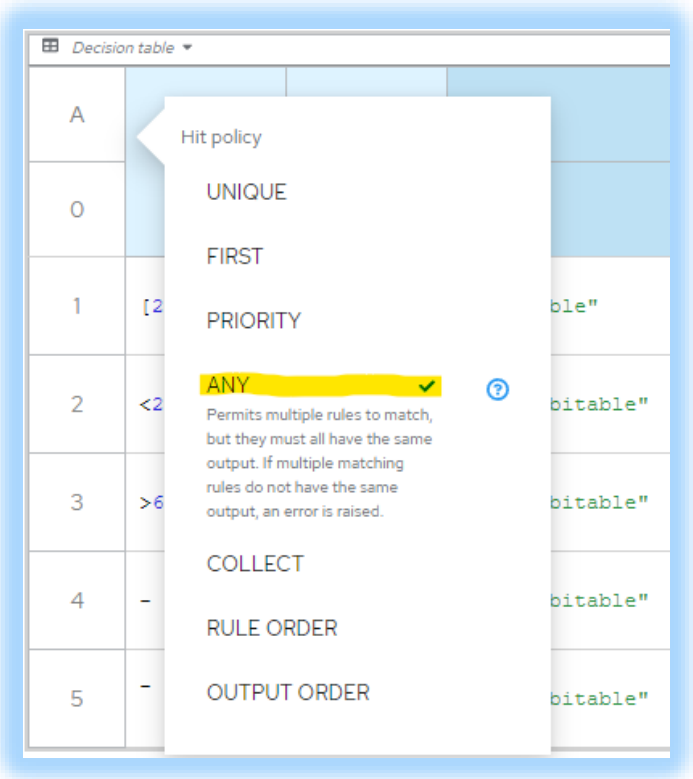
7. Click in the *Problems* button in the bottom right. There are several gap warnings:



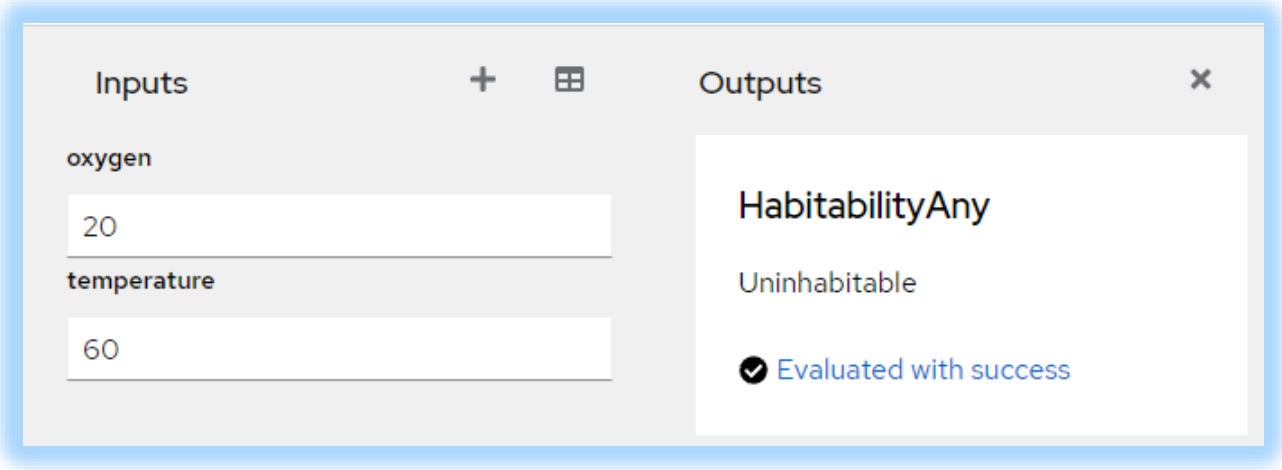
Validation 5

- DECISION_TABLE_HITPOLICY_RECOMMENDER: Overlapping rules have the same output value, so the HitPolicy for decision table 'HabitabilityAny' should be ANY
- DECISION_TABLE_OVERLAP_HITPOLICY_UNIQUE: Overlap detected: Overlap values: [<20, <0] for rules: [2, 5]. UNIQUE hit policy decision tables can only have one matching rule.
- DECISION_TABLE_OVERLAP_HITPOLICY_UNIQUE: Overlap detected: Overlap values: [<20, >50] for rules: [2, 4]. UNIQUE hit policy decision tables can only have one matching rule.
- DECISION_TABLE_OVERLAP_HITPOLICY_UNIQUE: Overlap detected: Overlap values: [>60, <0] for rules: [3, 5]. UNIQUE hit policy decision tables can only have one matching rule.
- DECISION_TABLE_OVERLAP_HITPOLICY_UNIQUE: Overlap detected: Overlap values: [>60, >50] for rules: [3, 4]. UNIQUE hit policy decision tables can only have one matching rule.

8. These errors a due to the Hit policy being **Unique** and there are overlapping rows. Fix these errors by changing the Hit Policy to **Any**. You should see the policy symbol change to **A** and all errors disappear:



9. Test by pressing R*un*. Enter Oxygen *20* and Temperature *60*:



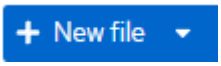
10. This tests row 4 of the decision table.

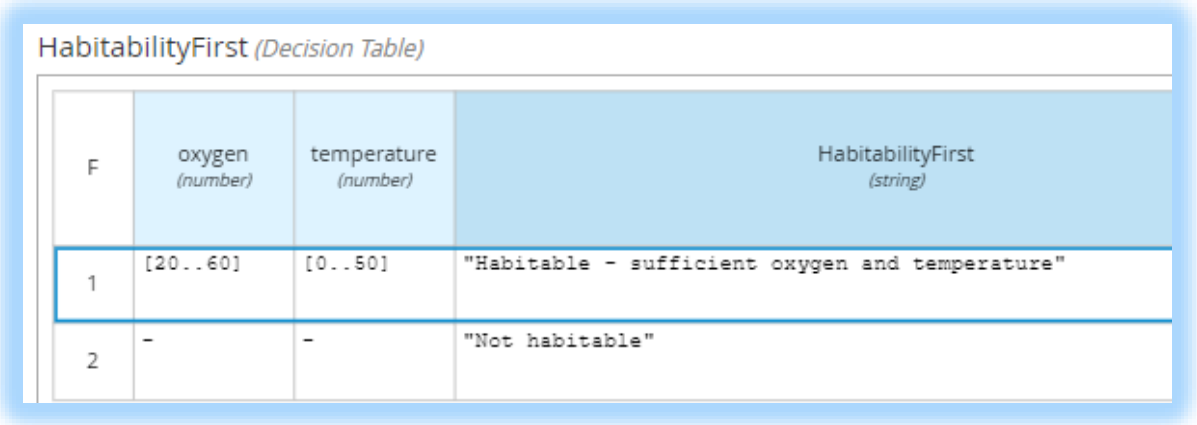
Conclusion

The advantage of the **Any** policy is you can create overlapping rules using "don't care" (-) values to make it more compact.

This advantage is also a disadvantage of imprecision; row 4 is hit whether oxygen is *habitable* or *uninhabitable*. This may be a problem if you need to know the exact reason as to why this planet is uninhabitable.

First Policy

1. In a new Chrome or Safari browser open the web site https://sandbox.kie.org/
2. If you already have KIE Sandbox open from the previous lab, clear the cache as recommended in Appendix A.
3. Click on **New Decision**.
4. An empty canvas opens. Click *New file*  and then U*pload...*
5. Select the file: lab05/*HabitabilityFirst.dmn*
6. Edit the *HabitabilityFirst* decision by clicking it and selecting Edit
7. You should see this:

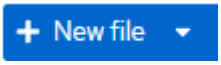


Test the decision by pressing *run*. Enter Oxygen *20* and Temperature *25* to check that row 1 fires.

Conclusion

Using the **First** hit policy you get the same result as the *Unique* and *Any* Policies, but with fewer rows. There are just two rows, one for a habitable planet, and another for an uninhabitable planet. The advantage is conciseness, but there are disadvantages. The first is that order matters – you cannot move row 2 to row 1. The **First** policy also has similar problems as the **Any** policy in that the decision has no detailed reasoning. Finally, the policy is the least strict, allowing overlaps and missing rows which could cause problems at run time.

String Collection policy

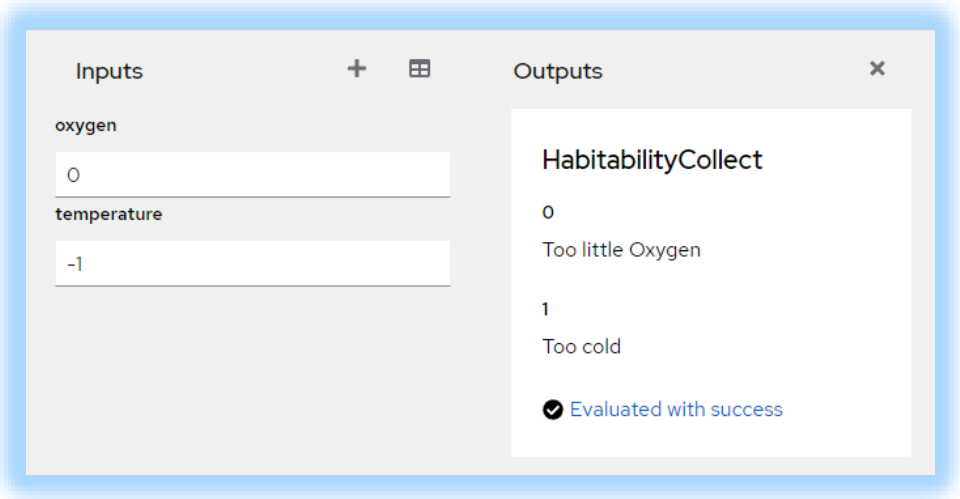
1. In a new Chrome or Safari browser open the web site https://sandbox.kie.org/
2. Click on **New Decision**.
3. An empty canvas opens. Click *New file*  and then U*pload…*

4. Select the file: *Lab05/HabitabilityStringCollect.dmn*
5. Edit the *HabitabilityCollect* decision by clicking it and selecting Edit
6. You should see this:

Decision table

C	oxygen (number)	temperature (number)	HabitabilityCollect (string)
0			HabitabilityCollect (string)
1	[20..60]	[0..50]	"Habitable - sufficient oxygen and temperature"
2	<20	-	"Too little Oxygen"
3	>60	-	"Too much Oxygen"
4	-	>50	"Too hot"
5	-	<0	"Too cold"

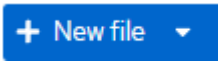
7. Test the decision by pressing *run*. Enter Oxygen *0* and Temperature *-1*. This time two rows are fired: row 2 and row 5. Both outputs are passed out of the decision as a list. Row 2 is passed as position 0 and row 5 as position 1 in the list.



Conclusion

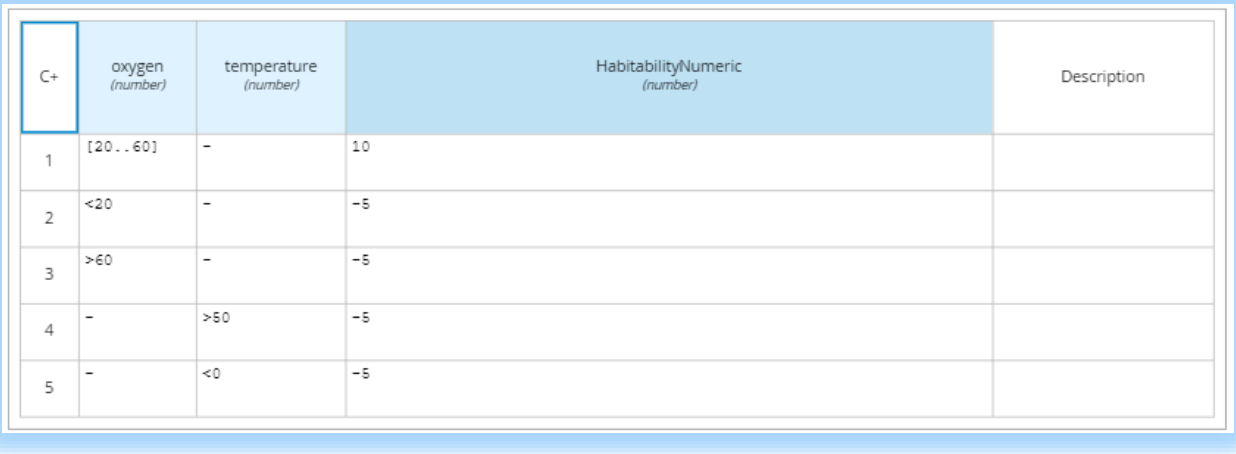
The Collection policy combines rows to make amalgamated decisions. This is useful when you require multiple rows in a decision table to be fired.

Numeric Collection policy

1. In a new Chrome or Safari browser open the web site https://sandbox.kie.org/
2. Click on **New Decision**.
3. An empty canvas opens. Click *New file* 

and then U*pload…*
4. Select the file: *Lab05/HabitabilityNumericCollect.dmn*

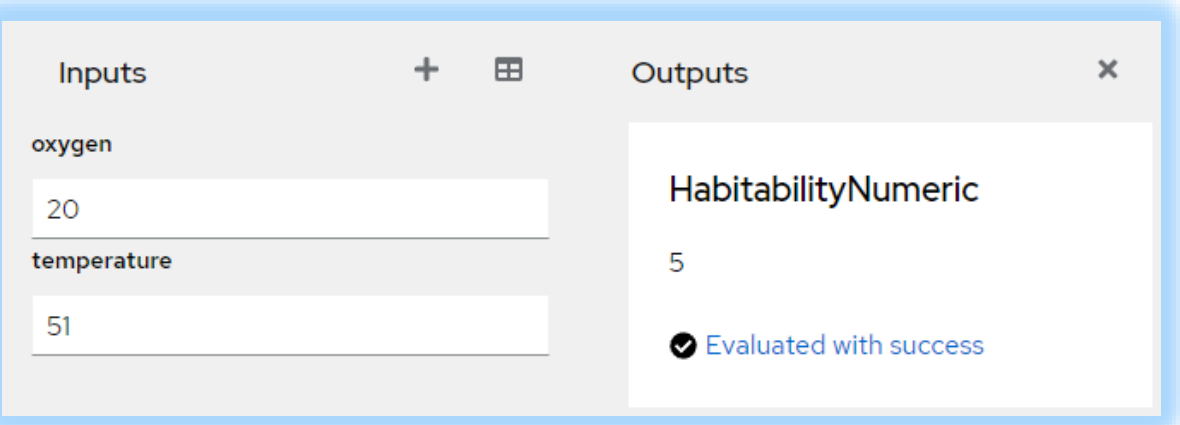
8. Edit the *HabitabilityNumeric* decision by clicking it and selecting Edit
9. You should see this:



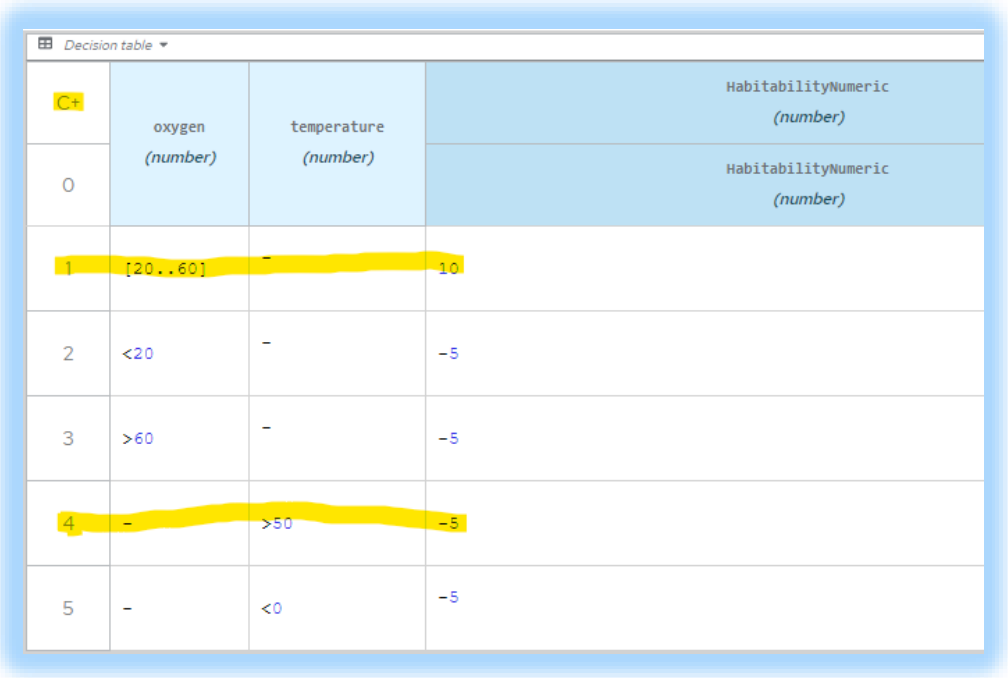
C+	oxygen *(number)*	temperature *(number)*	HabitabilityNumeric *(number)*	Description
1	[20..60]	-	10	
2	<20	-	-5	
3	>60	-	-5	
4	-	>50	-5	
5	-	<0	-5	

10. In the top left of the table, select the hit policy. Click the *C+* symbol. This symbol expands to show the Hit Policy *Collect SUM*:

11. The *Collect SUM* hit Policy collects all the rows satisfying the input conditions and sums them together. Let's test this. Press *Run*. Enter Oxygen *20* and Temperature *51*:



The *collect sum hit policy* collects rows 1 and 4 and then adds them together resulting in 5. See workings highlighted below:



Now run the test with following Hit Policies:

Symbol	Hit Policy / Aggregator	Result	**Comment**
C#	Collect/Count	**2**	**Counts the hits**
C<	Collect/Min	**-5**	**Returns the lowest value**
C>	Collect/Max	**10**	**Returns the highest value**

See the test results change for each aggregator. What are the applications for each aggregator?

Conclusion

In this lab we looked at hit policies. The choice of hit policy depends on whether you need simple *go/no go* decisions, or comprehensive reasoning.

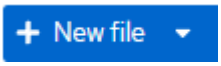
To continue with the next lab, please reset your environment as described in Appendix B.

Lab 6 - Advanced DMN

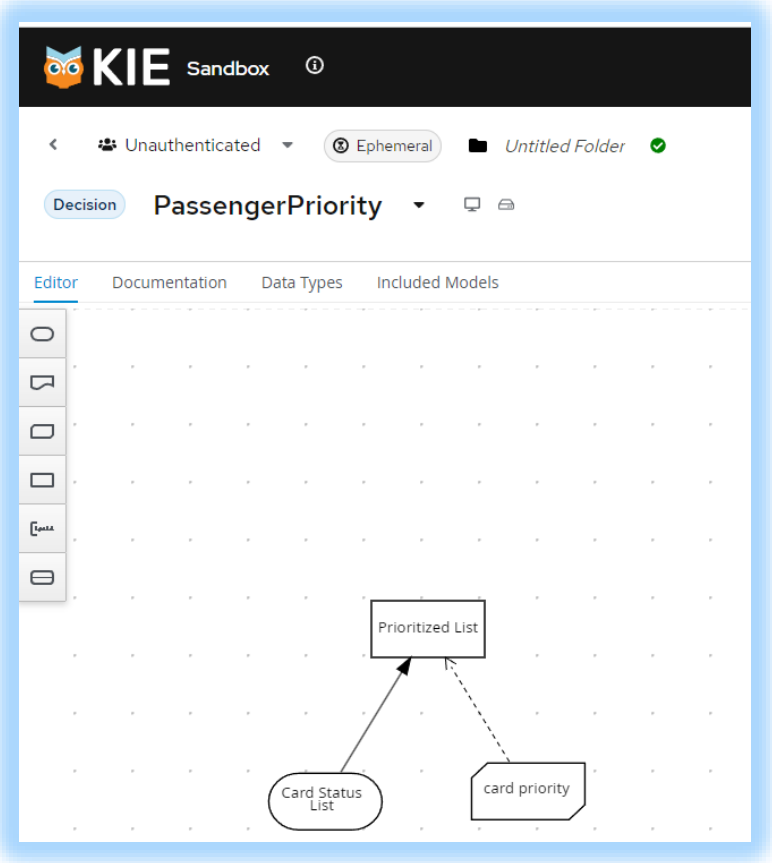
Introduction

Decision Requirements Diagrams and Decision Tables are something business users can understand. But it is difficult to do anything useful with these, except provide requirements to developers. The real power of DMN comes with FEEL (Friendly Enough Expression Language). In this lab we will explore an advanced FEEL example that reschedules flights.

Instructions

1. In a new Chrome or Safari browser open the web site https://sandbox.kie.org/
2. Click on **New Decision**.
3. An empty canvas opens. Click *New file* 

and then U*pload...*
4. Select the file: *Labs\Lab06\src\main\resources\Lab06\PassengerPriority.dmn*
5. You should see the following:



A Quick Tour of the Passenger Priority Service

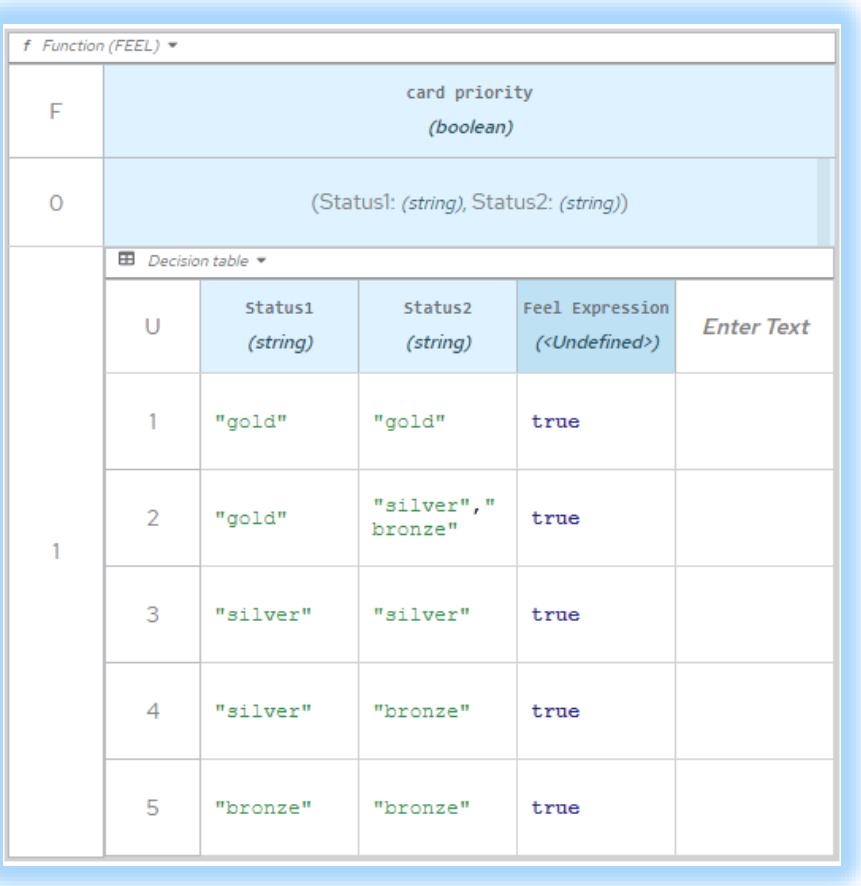
We will briefly examine the main components of the Passenger Priority service.

Click on the *data Types* tab. You should see:

- **CardStatusList** - a list of cards with status of either *gold, silver,* or *bronze*

12. Back in the DMN diagram, edit *card priority* by clicking it and selecting Edit

13. You should see this:



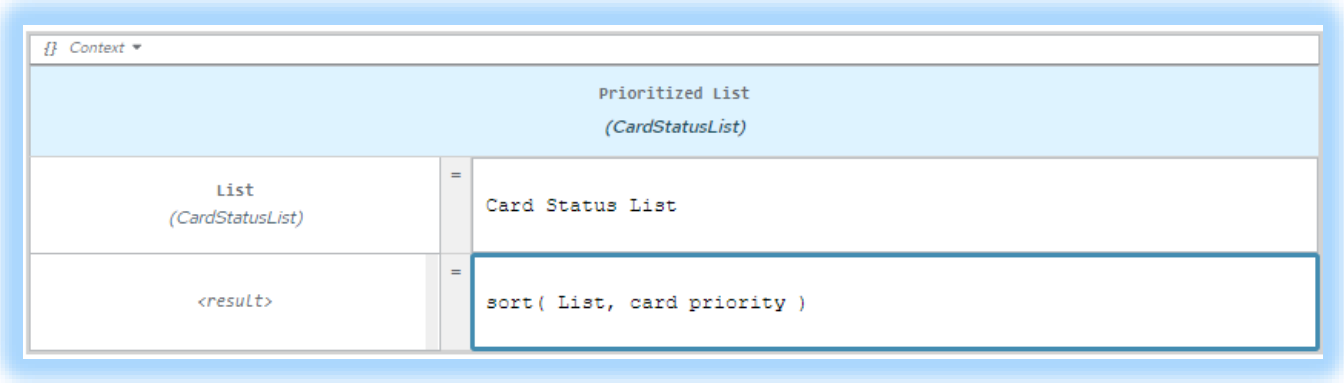
f Function (FEEL)

F	card priority (boolean)			
0	(Status1: *(string)*, Status2: *(string)*)			
1	Decision table			
	U	Status1 (string)	Status2 (string)	Feel Expression (<Undefined>)	Enter Text
	1	"gold"	"gold"	true	
	2	"gold"	"silver","bronze"	true	
	3	"silver"	"silver"	true	
	4	"silver"	"bronze"	true	
	5	"bronze"	"bronze"	true	

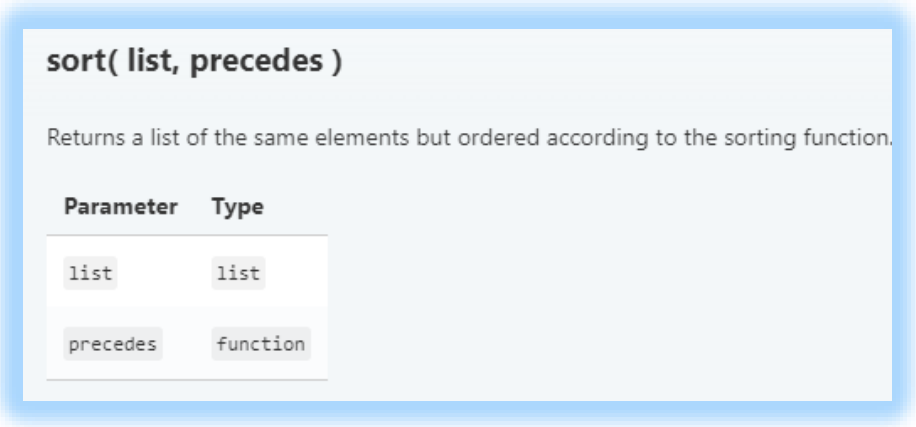
You will notice that it looks different to a standard decision table. It is a *Business Knowledge Model* which is a function invoked from a Decision. It takes parameters *Status1* and *Status2* which are of type String.

14. Back in the DMN diagram, edit the *Prioritized List* by clicking it and select Edit

15. You should see this:



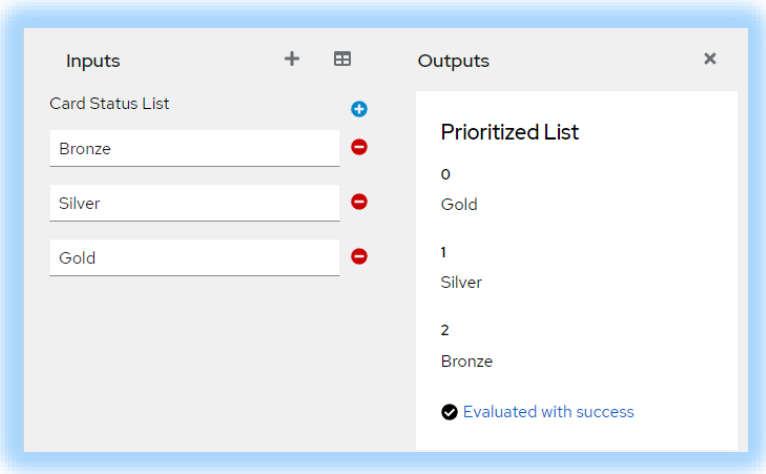
Note the highlighted *sort* function, which takes a list of cards . It returns the same list of passengers, but the list is sorted using the *card priority function* described above. The *sort* function is a built in FEEL function and is shown below:



Full details found here https://kiegroup.github.io/dmn-feel-handbook/#sort-functions

Test the Service

16. Press the *Run* button. In the inputs, add a random list of *gold, silver, and bronze* values. You should see the list sorted:

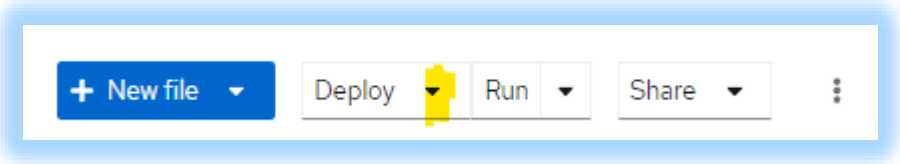


3 Conclusion

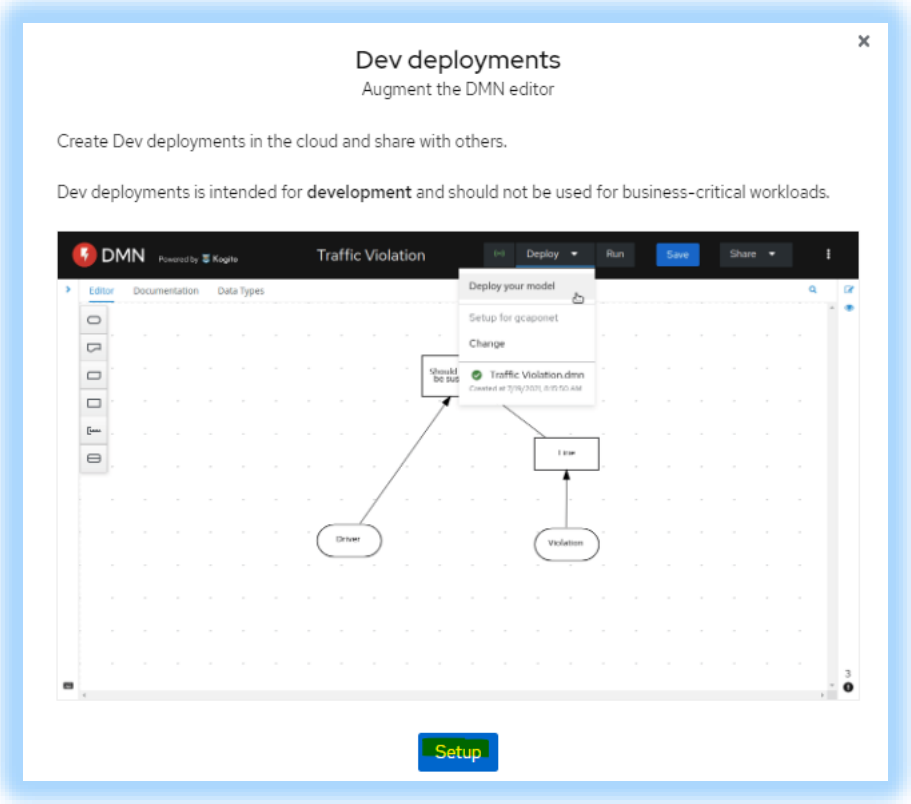
In this lab we went beyond the basics of DMN to give techniques for building real world DMN projects.

4 Appendix A: Installing KIE Sandbox Extended Services

In KIE Sandbox, click the arrow to the right of *Deploy*:



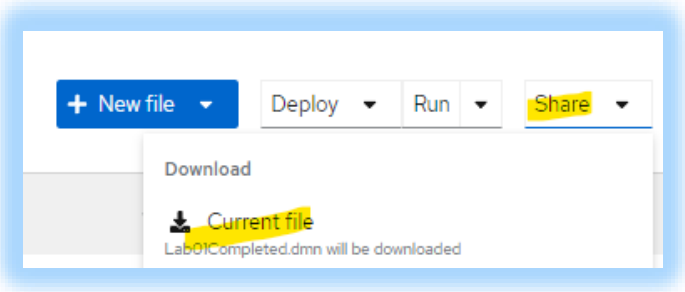
You should see the following screen. Click *Setup* and follow the on-screen instructions.



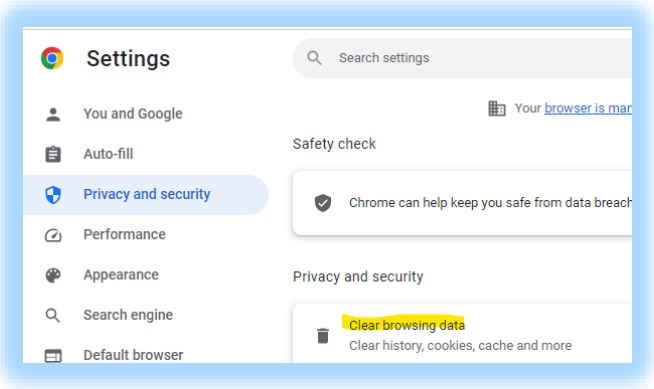
5 Appendix B: Clearing the KIE Sandbox Cache

If you are starting a new lab or the KIE Sandbox stops working, you will need to clear the cache.

You can save your work first by selecting *Share->Current File*. This will save your work to your local download directory. You can re-import this file after clearing the cache.



To clear the cache in Chrome, click ⋮ then select *Settings*. Select *Privacy and security* and press *Clear browsing data:*



As a last resort, close the browser completely and restart it. If you forgot to save your work, each lab has a completed DMN for you to see it in its final working state.