The Corticon Rule Modeling Methodology

A Case Study

CRM Use Case

By
Mike Parish

Contents

The Use Case 3

Basic Rule Modeling 3

- Identify the Business Decision(s) to be made 3
- Collect and Review Rules needed for each decision 3
- Identify Business Objects (Entities) 4
- Create Rule Sheet 4
- Create Test Sheet 6
- Enhancing the Model 6
- Create Natural Language Entries 9
- Organizing the Rules 9
- Exercise: 10

The Use Case

Use Corticon Business Rules to review accounts and create a flag record for each one that meets certain filter criteria.

Other tools are then used to read and process the flagged records

Basic Rule Modeling

Identify the Business Decision(s) to be made

Should this account be flagged?

Collect and Review Rules needed for each decision

The full set of rules is documented in the spreadsheet CRM_Filter List.xlsx

Here are the first four rules (out of a total of 108)

Category	Filter Name	Criteria
Loan Status	1st Lien Unusual String -- REO Reversal	All of (Lien Position = 1, Current Valuation Type ≠ "Sale Price", LIPD Not Advancing >= 4, Any of (Unusualstring R369F = "R3", Unusualstring R369F = "R6", Unusualstring R369F = "R9", Unusualstring R369F = "RF"), Any of (Signed CRMA = TRUE, Has Null CRMA? = 1))
Loan Status	1st Lien Unusual String-- C9	All of (Any of (Unusual string = "C9", Unusual string = "CF", Unusual string = "CR"))
Loan Status	Foreclosure REO with Current LIPD	All of (Delinquency Status in "F", "R", Any of (Signed CRMA = TRUE, Has Null CRMA? = 1), Months since LIPD <= 2)
Loan Status	Large Balance Decrease	All of (Delinquency Status in "3", "6", "9", "F", "R", Any of (Signed CRMA = TRUE, Has Null CRMA? = 1), All of (Unpaid Principal Balance > 1, Prior Month Unpaid Principal Balance > 1), Balance change percentage(m-m) > 50)

Let's take just the first rule and highlight the attributes that are being tested in red and the values in blue:

Category: Loan Status

Name: 1st Lien Unusual String -- F99

Filter Criteria:

All of (Lien Position = 1, Delinquency Status = "9", Time Not In Bankruptcy >= 3, Time Not In Payment Plan >= 3, LIPD Not Advancing >= 4, Unusualstring = "F99", Signed CRMA = TRUE, Loss Mitigation Flag = FALSE)

Identify Business Objects (Entities)

The only object at this point appears to be the Account which is being reviewed.

Using the attributes highlighted in red above we can create a simple vocabulary that supports this rule (we'll add more attributes as required by other rules later)

This leads us to a starting vocabulary like this:

- Account
  - delinquencyStatus
  - lienPosition
  - LIPDNotAdvancing
  - lossMitigationFlag
  - signedCRMA
  - timeNotInBankruptcy
  - timeNotInPaymentPlan
  - unusualString

Create Rule Sheet

We start by creating a new rule sheet and pasting the rule definition into the rule statement section.

In order to make it a little more readable we can use SHIFT+ENTER to insert line breaks.

We can also add the rule name and category to the rule statement:

Text	Rule Name	Source Name	Category	Comments
All of (Lien Position = 1, Delinquency Status = "9", Time Not In Bankruptcy >= 3, Time Not In Payment Plan >= 3, LIPD Not Advancing >= 4, Unusualstring = "F99", Signed CRMA = TRUE, Loss Mitigation Flag = FALSE)	1st Lien Unusual String -- F99	CRM Filter_List.xlxs	Loan Status	

We'll use this to guide the process of modeling the rules.

Now we can drag the various attributes into the condition section of the rule sheet:

CRM.ers

	Conditions	1	2
a	Account.lienPosition		
b	Account.delinquencyStatus		
c	Account.timeNotInBankruptcy		
d	Account.timeNotInPaymentPlan		
e	Account.LIPDNotAdvancing		
f	Account.unusualString		
g	Account.signedCRMA		
h	Account.lossMitigationFlag		
i			

Next we enter in column 1 the values against which each of these attributes is to be tested:

CRM.ers

	Conditions	1
a	Account.lienPosition	1
b	Account.delinquencyStatus	'9'
c	Account.timeNotInBankruptcy	>= 3
d	Account.timeNotInPaymentPlan	>= 3
e	Account.LIPDNotAdvancing	>= 4
f	Account.unusualString	'F99'
g	Account.signedCRMA	T
h	Account.lossMitigationFlag	F
i		

Finally we create an appropriate message in the rule statement section and attach it to rule column 1 using the REF column:

*CRM.ers

	Conditions	1	2
a	Account.lienPosition	1	
b	Account.delinquencyStatus	'9'	
c	Account.timeNotInBankruptcy	>= 3	
d	Account.timeNotInPaymentPlan	>= 3	
e	Account.LIPDNotAdvancing	>= 4	
f	Account.unusualString	'F99'	
g	Account.signedCRMA	T	
h	Account.lossMitigationFlag	F	
i			
	Actions		
	Post Message(s)		
A			
	Overrides		

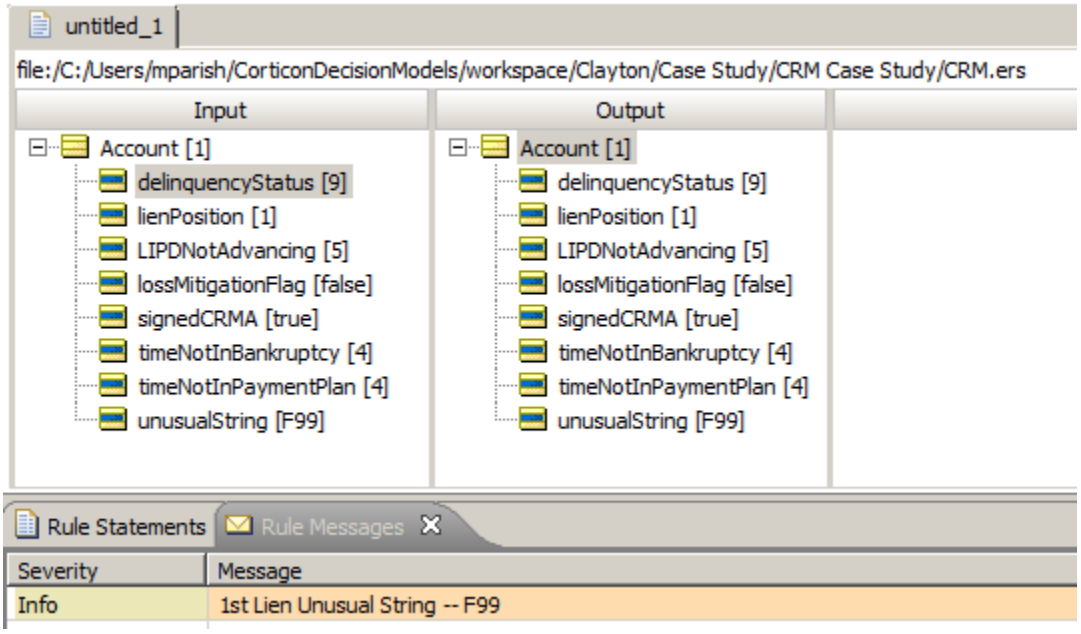
Rule Statements

Ref	Post	Alias	Text	Rule Name
			All of (Lien Position = 1, Delinquency Status = "9", Time Not In Bankruptcy >= 3, Time Not In Payment Plan >= 3, LIPD Not Advancing >= 4, Unusualstring = "F99", Signed CRMA = TRUE, Loss Mitigation Flag = FALSE)	1st Lien Unusual String -- F99
1	Info	Account	1st Lien Unusual String -- F99	

Notice we have marked this as an informational message and that it belongs to the Account.
We've chosen to display a message that shows the name of the filter that applied.
It is also possible to attach the entire rule definition as a message. However if you want to do this you must remove the SHIFT+ENTER (otherwise you will get a compile error)

Create Test Sheet

Now we can create a simple test case to make sure this filter works:



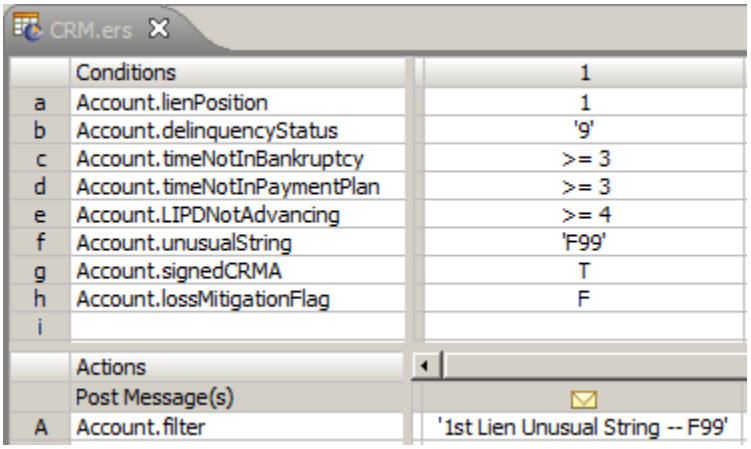
The rule message indicates which filter applied.

Enhancing the Model

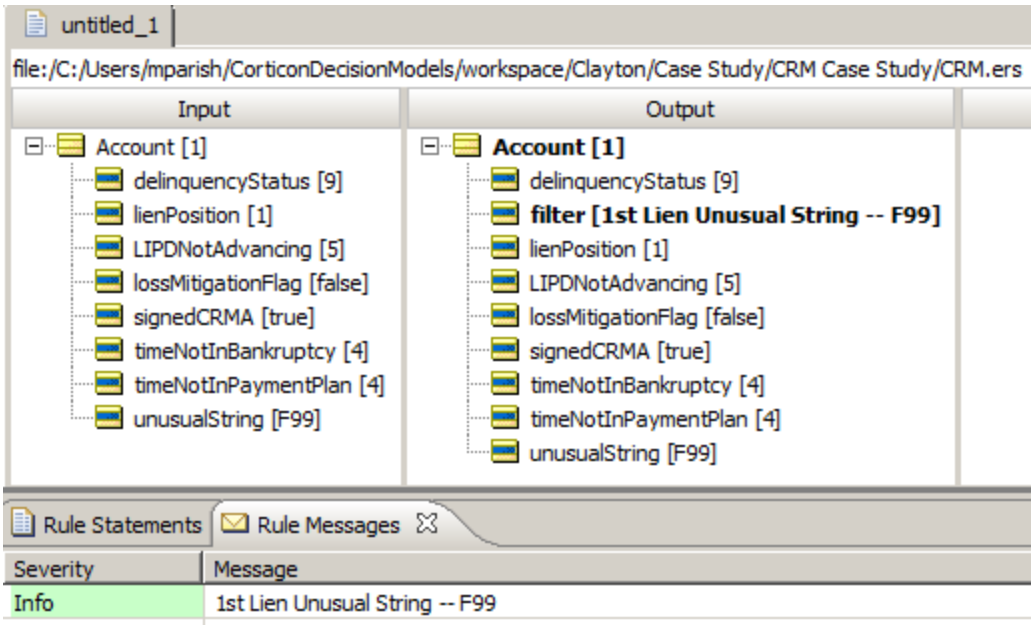
As it stands, the only place we will see which filter applied is in the list of messages.

Let's modify the rules so the filter is saved into the Account for further use:

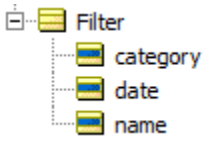
We'll first need to add an attribute to the vocabulary for the filter and then we can update the rule sheet as follows:



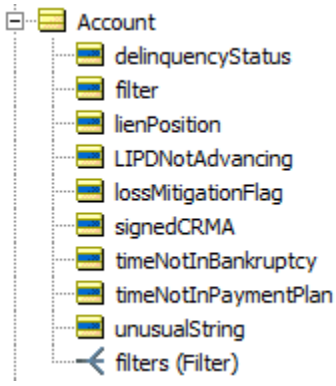
And the test results look like this:



But there's a potential problem with this approach.
Right now we only have one rule. But what if there are many rules and several of them could apply?
We would find that the filter attribute would be overwritten each time an applicable rule fired.
The key to solving this is recognizing that we are dealing with a one-to-many relationship between an Account and the many filters that could apply to it.
One simple way to deal with it is to concatenate the filter name into the filter attribute (separated by commas).
A better way is to create a new entity specifically for dealing with the filters. This now gives us the ability to record lots of information rather than just the name.
The entity could look like this:



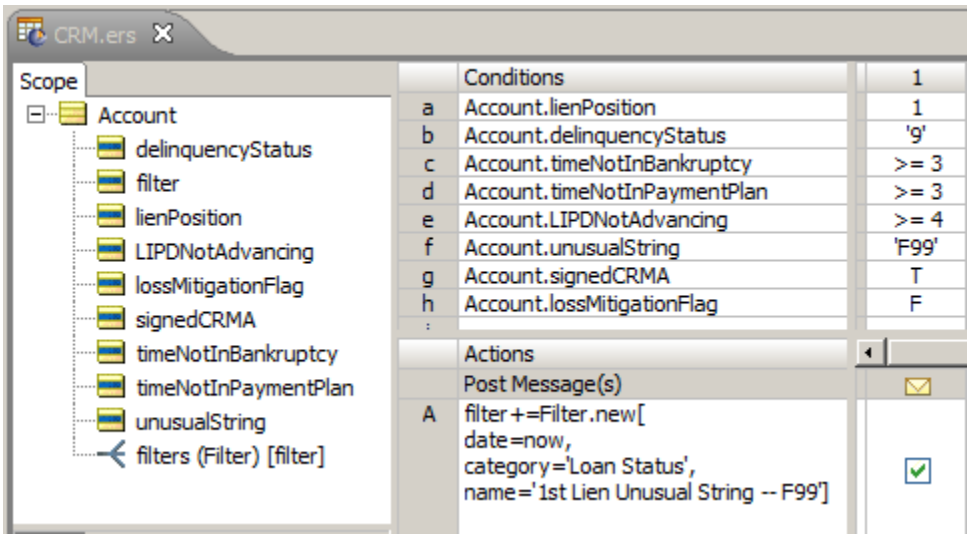
Then we can show that there is a one-to-many relationship between an Account and applicable Filters as follows:



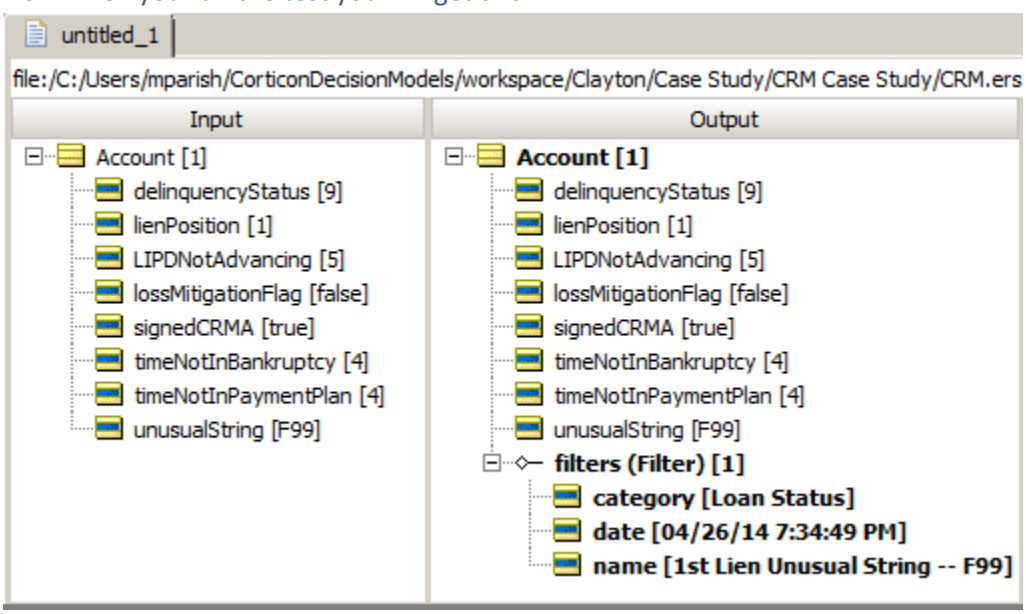
Next we need to update the rule so that it will create this new Filter object whenever a rule fires. There are two steps to this:

1. Define aliases in the Scope Section
2. Use the new operator in the action of the rule

Here's the result



Now when you run the test you will get this:



Now when you add more rules you will see multiple occurrences of the filter object if more than one filter applies.

Create Natural Language Entries

To make the rule sheet even more readable we can enter natural language forms for each of the conditions and actions.

For example:

CRM.ers

	Conditions	1
a	Lien Position	1
b	Delinquency Status	'g'
c	Time not in Bankruptcy	>= 3
d	Time not in Payment Plan	>= 3
e	LIPD Not advancing	>= 4
f	Unusual String	'F99'
g	Signed CRMA	T
h	Loss Mitigation Flag	F
i		
	Actions	
	Post Message(s)	
A	Flag this account as meeting the 1st Lien Unusual String -- F99 filter	✓

Organizing the Rules

Although it's possible to put all the rules in a single rule sheet it may be better to divide them up into rule sheets with sets of related rules.

Since the spreadsheet groups the rules in categories (Loan Status, Mortgage Insurance etc) we'll do the same.

Once we have each of the rule sheets modeled, we can construct the decision service by dragging the rule sheets onto a rule flow diagram:

Exercise:

Using the spreadsheet as a guide:

1. add more rule statements
2. add attributes to the vocabulary if necessary
3. create the rule columns
4. add a test case for each rule you create