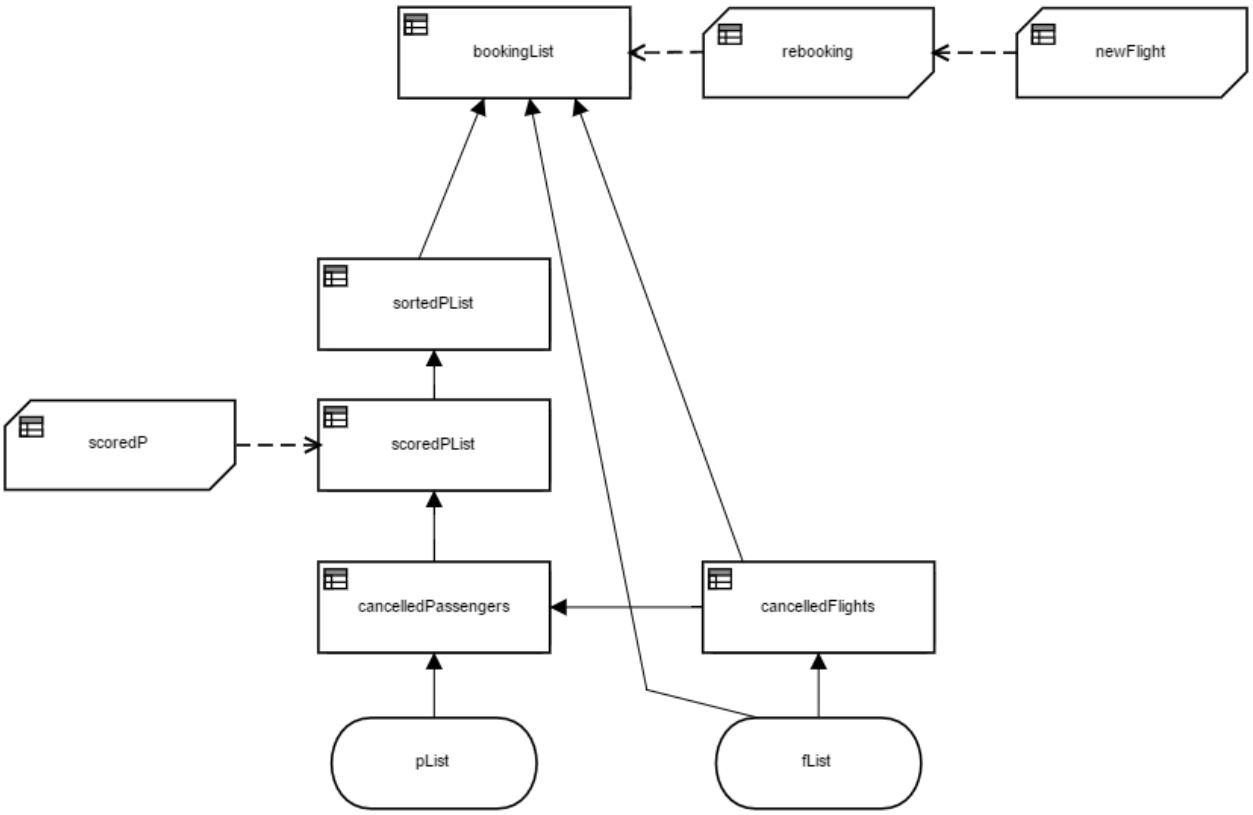
Passenger Rebooking Decision Modeling Challenge

Method and Style Solution, Bruce Silver

My solution to the challenge uses DMN 1.1. I suspect this may be the first publication demonstrating a complete DMN tool, including DRDs, full use of boxed expressions, and FEEL. The diagrams and tables were created using the Trisotech DMN Modeler, and the DMN execution is from Method and Style.

The solution provides a good illustration of the power of DMN, including its ability to manipulate lists and tables. It also exploits a previously unremarked (you might call it "hidden") feature of DMN, which is recursion.



The DRD describes the end-to-end solution. The input data elements pList and fList are the passenger list and flight list tables provided. For some reason they were already presorted by priority, but the solution does not depend on that.

cancelledFlights

cancelledFlights is a simple filter expression.

cancelledFlights (Decision)

Output Data Type

Type	tFList

Decision Logic (Boxed FEEL Expression)

cancelledFlights

```
fList[status="cancelled"]
```

cancelledPassengers

cancelledPassengers is a relational join between two tables, pList and cancelledFlights. If such a join selects a single item, it can be done with nested filters, but here that is not the case, so we use an iteration.

cancelledPassengers (Decision)

Output Data Type

Type	tPList

Decision Logic (Boxed FEEL Expression)

cancelledPassengers

```
for i in pList return (if cancelledFlights[fnum = i.flight] then i else null)
```

FEEL's for..in..return syntax is basically the same as XPath, if you are familiar with that.

sortedPList

Passengers are sorted for rebooking in priority order, priority determined by combination of status and miles. Presumably miles serve as a tiebreaker for passengers with same status. Since FEEL sort function has only a single sort key, I used a C+ decision table to compute a numerical score combining status and miles, and sorted on that. The decision scoredPList iterates the BKM scoredP, and then the decision sortedPList sorts scoredPList on the score component.

scoredPList (Decision)

Output Data Type

Type	tPList

Decision Logic (Boxed FEEL Expression)

scoredPList

```
for i in cancelledPassengers return scoredP(i)
```

scoredP (Business Knowledge Model)

Output Data Type

Type	tPassenger

Decision Logic (Boxed Function Context)

scoredP

(passenger*(tPassenger)*)

score	C+	passenger.status	passenger.miles	score
		Text	*Number*	*Number*
	1	"Gold"	-	100000
	2	"Silver"	-	25000
	3	"Bronze"	-	10000
	4	-	-	passenger.miles

scoredPassenger *(tPassenger)*	name	passenger.name
	flight	passenger.flight
	score	score

```
scoredPassenger
```

sortedPList (Decision)

Output Data Type

Type	tPList

Decision Logic (Boxed FEEL Expression)

sortedPList

```
sort(scoredPList, function(x,y) x.score>y.score)
```

bookingList

The bookingList decision invokes the BKM rebooking. rebooking is not a normal iteration but a recursion, meaning the BKM rebooks one passenger and then if there are remaining unbooked passengers it calls itself to book the next one. The reason I did it this way is that the available flights change with each booked passenger. rebooking has four parameters: unbooked, a list of unbooked passengers; rebooked, a list of rebookings; fList, the list of available flights; and originalFList, the original flight list. When it is invoked the first time, unbooked is the sorted list of passengers and rebooked is an empty list.

bookingList (Decision)

Output Data Type

Type	tBookingList

Decision Logic (Boxed FEEL Expression)

bookingList

```
rebooking(sortedPList,[],fList,fList)
```

rebooking (Business Knowledge Model)

Output Data Type

Type	tBooking

rebooking is a context. thePassenger is the first one in the unbooked list. Other context entries gather that passenger's originalFlight, its departure time (since the rebooking must depart after that time), and its destination (since the rebooking must have the same destination). availableFlights is a filter selecting from the parameter fList the flights not cancelled that have the same destination and remaining seats available.

firstArrival selects from availableFlights the earliest arrival time, and bookedFlight selects the flight matching that time. Note the odd syntax of firstArrival, since the min semantics depends on the datatype (number, string, dateTime, etc.), so we need to convert arrive, a string, to a dateTime element. newBooking is a nested context, containing the passenger name, rebooked flight, and arrival time. newRebooked appends the new booking to the rebooked list, and newUnbooked removes the first item from the old unbooked list. newFlightList iterates through availableFlights and deducts one available seat from the flight just rebooked. bookings then either recurses or – if no remaining unbooked – exits. bookings is the final result box of the BKM, the value reported back to bookingList.

rebooking		
(unbooked*(tBookingList)*, rebooked*(tBookingList)*, fList*(tFList)*, originalFList*(tFList)*)		
thePassenger *(tPassenger)*	`unbooked[1]`	
originalFlight *(Text)*	`originalFList[fnum=thePassenger.flight]`	
originalDepart *(Date and time)*	`originalFlight.depart`	
theDestination *(Text)*	`originalFlight.to`	
availableFlights *(tFList)*	`fList[status="scheduled" and to=theDestination and seatsAvailable!=0]`	
isFlightAvailable *(Boolean)*	`if count(availableFlights)>0 then true else false`	
firstArrival *(Date and time)*	`min(availableFlights.date and time(arrive))`	
bookedFlight *(tFlight)*	`availableFlights[arrive=firstArrival]`	
newBooking *(tBooking)*	name *(Text)*	`thePassenger.name`
	flight *(Text)*	`if isFlightAvailable=true then bookedFlight.fnum else "none"`
	arrive *(Date and time)*	`if isFlightAvailable=true then firstArrival else "-"`
newRebooked *(tBookingList)*	`append(rebooked,newBooking)`	
newUnbooked *(tBookingList)*	`remove(unbooked,1)`	
newFlightList *(tFList)*	`for i in availableFlights return newFlight(i,bookedFlight)`	
bookings *(tBookingList)*	`if count(newUnbooked)>0 then rebooking(newUnbooked,newRebooked,newFlightList) else newRebooked`	
`bookings`		

newFlight (Business Knowledge Model)

Output Data Type

Type	tFlight

Decision Logic (Boxed Function Context)

newFlight		
(flight*(tFlight)* , bookedFlight*(tFlight)*)		
newSeatsAvailable *(Number)*	`if flight=bookedFlight then flight.seatsAvailable -1 else flight.seatsAvailable`	
updatedFlight *(tFlight)*	fnum	`flight.fnum`
	from	`flight.from`
	to	`flight.to`
	depart	`flight.depart`
	arrive	`flight.arrive`
	seatsAvailable	`newSeatsAvailable`
	status	`flight.status`
`updatedFlight`		

The datatypes (itemDefinitions) are shown below, followed by execution results. The execution result tables are generated directly via xslt from the XML output of DMN execution.

Datatypes

name	base
tPList	tns:tPassenger (isCollection=true)
tFList	tns:tFlight (isCollection=true)
tBookingList	tns:tBooking (isCollection=true)
tPassenger	complex type
* name	feel:string
* status	feel:string
* miles	feel:number
* flight	feel:string
* score	feel:number
* index	feel:number
tFlight	complex type
* fnum	feel:string
* from	feel:string
* to	feel:string
* depart	feel:dateTime
* arrive	feel:dateTime
* seatsAvailable	feel:number
* status	feel:string
tBooking	complex type
* name	feel:string
* flight	feel:string
* to	feel:string
* arrive	feel:dateTime

Inputs add @id

pList

name	status	miles	flight				
Harry	Gold	100000	UA123				
Igor	Gold	50000	UA123				
Jenny	Gold	500000	UA123				
Dick	Silver	100	UA123				
Tom	Bronze	10	UA123				

fList

fnum	from	to	depart	arrive	seatsAvailable	status	
UA123	SFO	SNA	2007-01-01T18:00:00	2007-01-01T19:00:00	5	cancelled	
UA456	SFO	SNA	2007-01-01T19:00:00	2007-01-01T20:00:00	2	scheduled	
UA789	SFO	SNA	2007-01-01T21:00:00	2007-01-01T23:00:00	2	scheduled	
UA1001	SFO	SNA	2007-01-01T23:00:00	2007-01-02T05:00:00	0	scheduled	
UA1111	SFO	LAX	2007-01-01T23:00:00	2007-01-02T05:00:00	2	scheduled	

Outputs

cancelledFlights

Computed							
fnum	from	to	depart	arrive	seatsAvailable	status	
UA123	SFO	SNA	2007-01-01T18:00:00	2007-01-01T19:00:00	5	cancelled	

cancelledPassengers

Computed							
name	status	miles	flight				
Harry	Gold	100000	UA123				
Igor	Gold	50000	UA123				
Jenny	Gold	500000	UA123				
Dick	Silver	100	UA123				
Tom	Bronze	10	UA123				

scoredPList

Computed							
name	flight	score					
Harry	UA123	200000					
Igor	UA123	150000					
Jenny	UA123	600000					
Dick	UA123	25100					
Tom	UA123	10010					

sortedPList

Computed							
name	flight	score					
Jenny	UA123	600000					
Harry	UA123	200000					
Igor	UA123	150000					
Dick	UA123	25100					
Tom	UA123	10010					

bookingList

Computed							
name	flight	arrive					
Jenny	UA456	2007-01-01T20:00:00					
Harry	UA456	2007-01-01T20:00:00					
Igor	UA789	2007-01-01T23:00:00					
Dick	UA789	2007-01-01T23:00:00					
Tom	none	-					