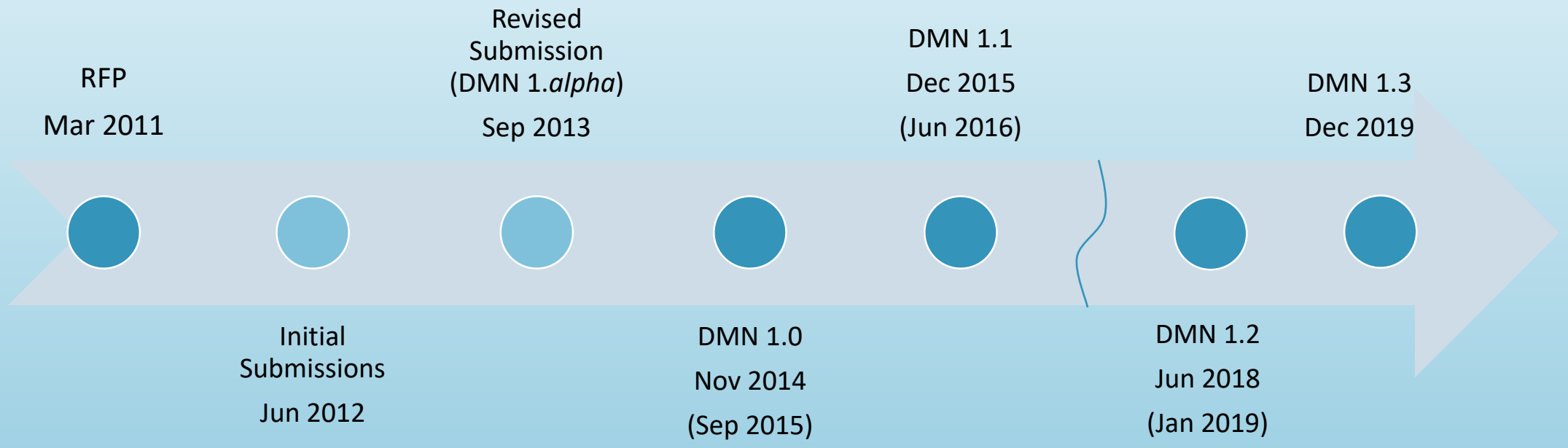
DMN 2.0?

Experience from DMN 1.0 – 1.3

Agenda

- Background
- Top 10 Requested Features
- Discussion

Background – History of DMN 1.x



Major and Minor Versions (*i.j*)

Major (*i*)

- Required for incompatible changes
- Recommended for major new features
- Must issue Request For Proposal
- Submit Proposal(s)
  - Requires OMG Submitter Membership
  - Competing submissions are possible
- Submitted document becomes new base for *2.0*, **not** the latest *1.j* document
- May be more efficient for major changes

Minor (*j*)

- Must be compatible
- Should fix bugs rather than add major new features
- Must carefully track and justify every change
- Output of Task Force is a detailed list of changes, **not** a new document
- Above also applies going from Revised Submission to *2.0*. Be sure your Revised Submission does not need major work (painful lesson)

Compatibility

- DMN *1.x* tool/engine correctly interprets a *1.y* decision model
- Backward compatible: *x >= y*
- Forward compatible: *x <= y*
- We do not provide forward compatibility

Top 10 Requested Features

Compatible with DMN 1.x

- Harmonize DMN, BPMN, and CMMN
- Standard Model Validation
- Better Error and Null Handling
- Additional Datatypes
- Better Iteration
- Recursion

Not Compatible

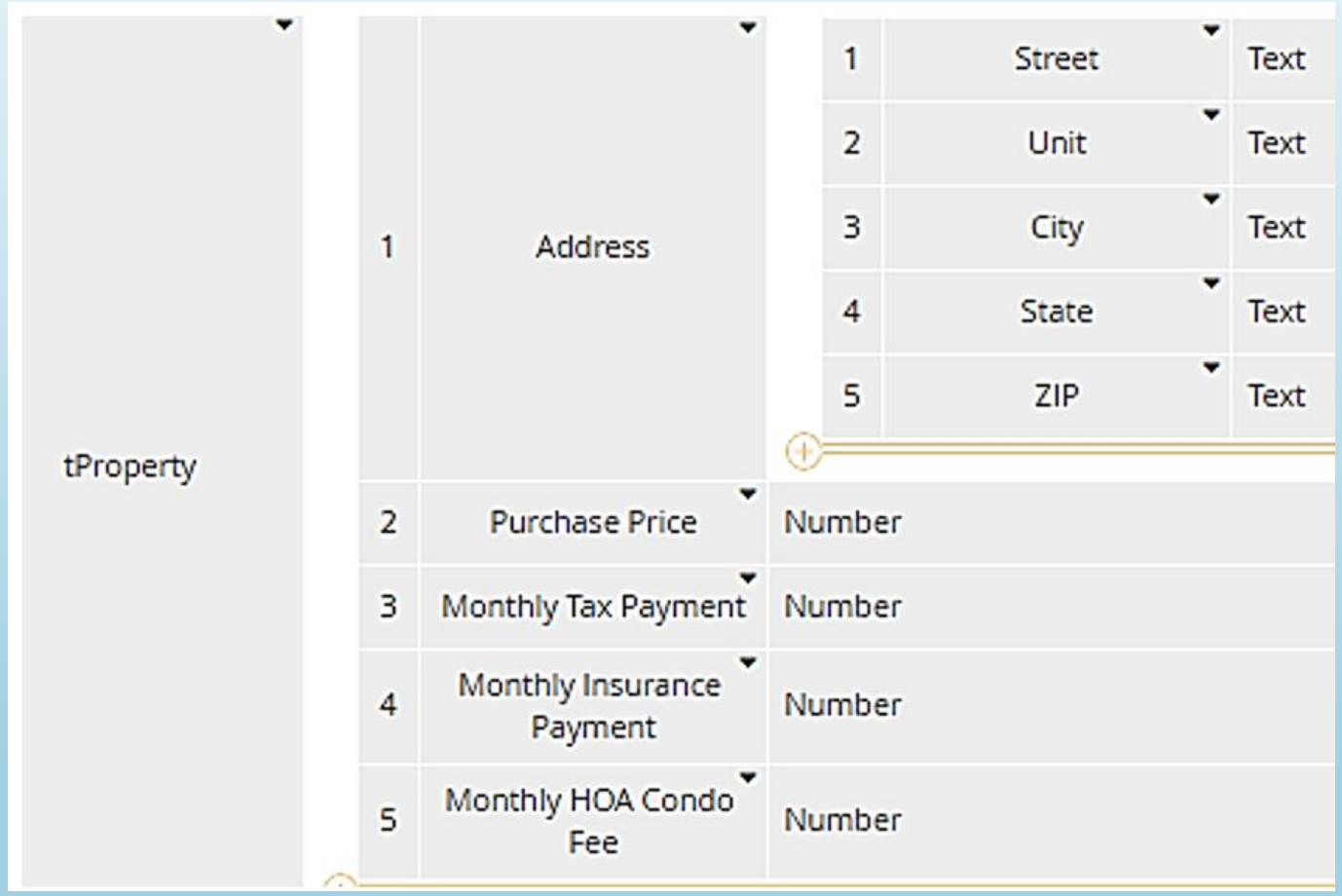
- Unambiguous Context-free grammar
- Case Insensitive Names
- Sequences instead of Lists
- Cyclic Information Requirements, Constraint Solving

Harmonize DMN, BPMN, and CMMN

- Decision, Business Process, and Case Management Modeling and Notation
- Common *Item Definition* model across the 3 standards
  - Graphical and/or tabular notation
- Make it easier for models to reference each other and be interchanged together
- Tighter integration of DMN with BPMN gateways and CMMN Sentries

Harmonize DMN, BPMN, and CMMN

Common
Item Definition
Model and Notation

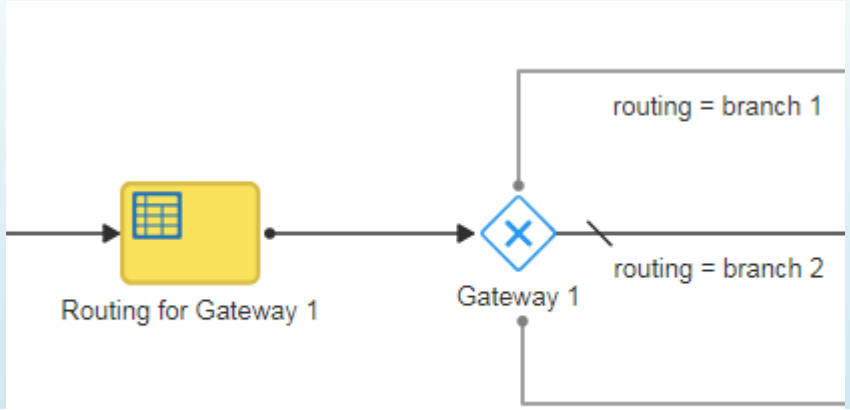


tProperty	1	Address	1	Street	Text
			2	Unit	Text
			3	City	Text
			4	State	Text
			5	ZIP	Text
	2	Purchase Price	Number		
	3	Monthly Tax Payment	Number		
	4	Monthly Insurance Payment	Number		
	5	Monthly HOA Condo Fee	Number		

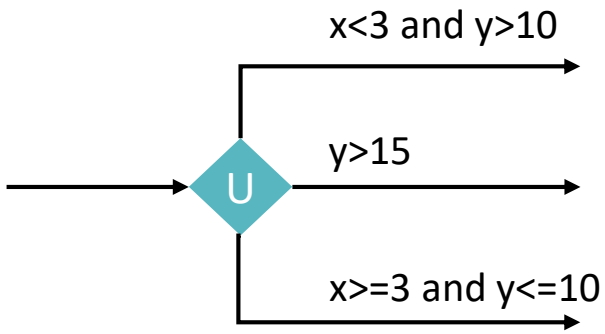
Harmonize DMN, BPMN, and CMMN

- Notation synergy
  - FEEL is the type and expression language
  - FEEL instead of comments on gateways and sentries
  - Gateway and sentry logic as decision tables to check for conflicts and gaps
  - FEEL for data associations, *e.g.* ***applicant.creditScore, applicant.salaryRange***
- Model synergy
  - All standards are model-driven with named shareable elements inside a top-level ***Definition***
  - Problems with Import
    - Each model must import every model it references – N^2 imports
    - Interchange may break the import references
  - Solution
    - Define a Model Archive to package multiple related models
    - Models in archive may reference each other without explicit imports
    - Models in archive are interchanged together

Harmonize DMN, BPMN, and CMMN



Surface DMN to organize gateway logic



Gateway 1

Decide which exit branch to take

Q&A

3

Validation Errors/Warnings

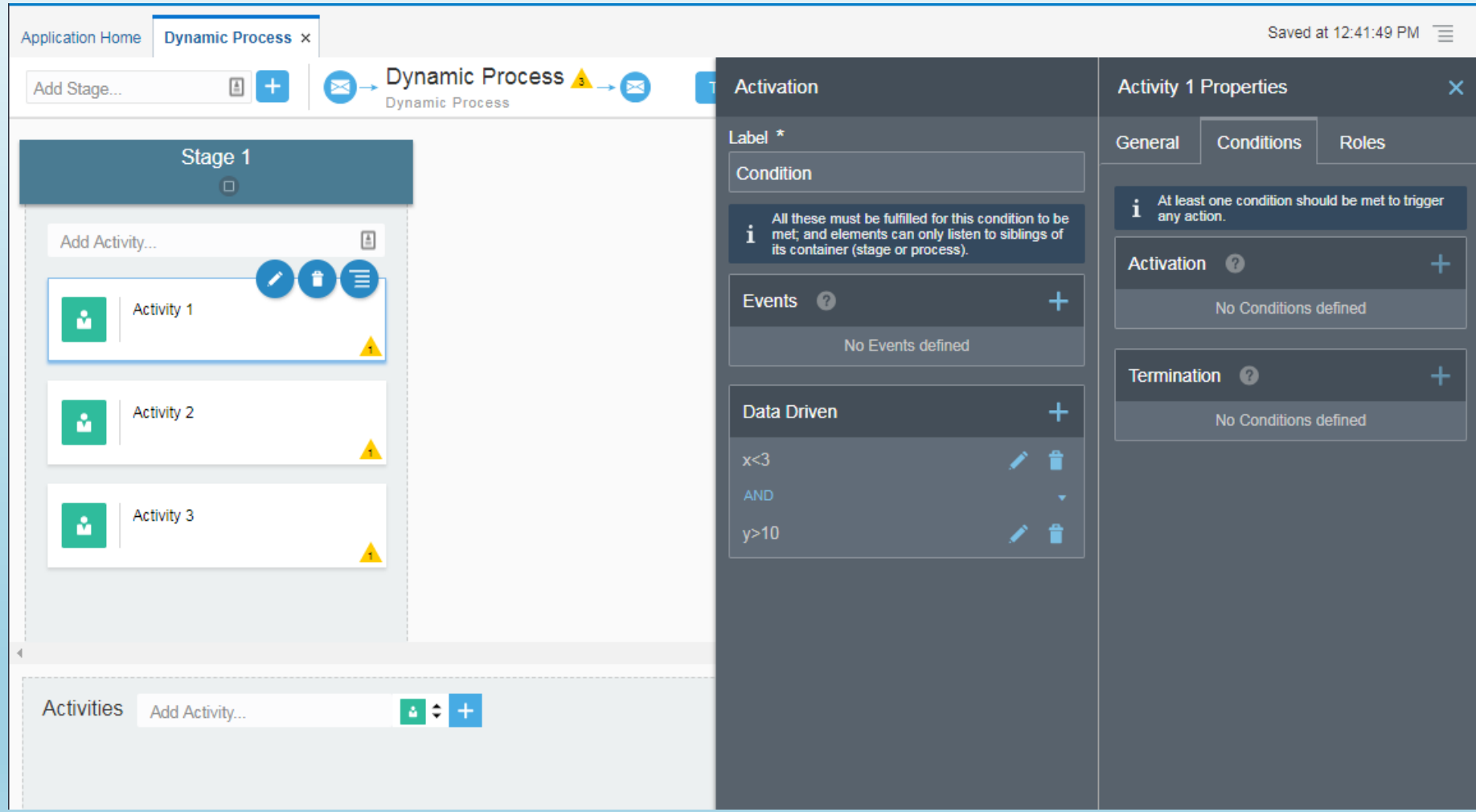
1 Missing rule with conditions: <3 and <=10
2 Missing rule with conditions: >=3 and (10..15]
3 Overlapping rules 1, 2

Decision Table

U	x -	y -	Gateway 1 *Enter Allowed Values*
1	<3	>10	*branch 1*
2	-	>15	*branch 2*
3	>=3	<=10	*branch 3*

Harmonize DMN, BPMN, and CMMN

- Use DMN to organize sentry logic in a case
- Validate that some activity or activities can always run (depending on hit policy)



Standard Model Validation

- FEEL Semantics are execution-oriented
- All errors result in **null**
- Many DMN tools validate a model before execution
- No standard set of validations or error codes are defined
- For example,
  - *X* and *Y* are known to be Boolean
  - We can infer that *X + Y* = **null**.
  - Should tools be required to report a model validation error?
  - Should it be standard?, *e.g.*, **DMN-001: addition cannot be applied to boolean**

Error and Null Handling

- Where does Null come from?
  - Missing input data, *e.g.,* Person.Gender
  - FEEL semantic errors, *e.g.* 1/0, true + 1, no rule fired
  - Explicit keyword **null**
- Null Propagation
  - If **null** is not explicitly allowed as input by a FEEL function or operator, then its output will be **null**
  - (Person.Gender="F") = null
  - string length(Person.Gender) = null
  - not(null) = null
  - sum(1, 2, null) = null
- Catch Null
  - Some operators can have a *non*-**null** result with a **null** input
  - (null = null) = true
  - [1, null, 2] [item != null] = [1, 2]
  - (if null then null else 2) = 2
  - (null and false) = false
  - (null or true) = true
- Do we need mnemonic syntax?
  - ***=null****
  - ***if****
  - ***and****
  - ***or****

Better Null Handling

New Item Definition Property

- **required** – expressions of this type may not be null
- ***2*** is a **required number**
- ***X + Y*** is a **required number** only if both ***X*** and ***Y*** are required numbers

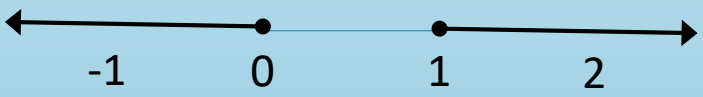
Model Validation

- Expression type and nullability inference
- Validation with respect to Item Definitions
- A warning must be given if inferred expression type and nullability do not agree with associated Item Definition (if any):
  Cannot require *X + Y* because *X* is not required

Additional Datatypes

Integers

A decision table with a number input and rules for *<=0* and *>=1* is incomplete, having omitted a rule for *(0..1),* whereas if the input is known to be an integer, the table is complete



Ranges

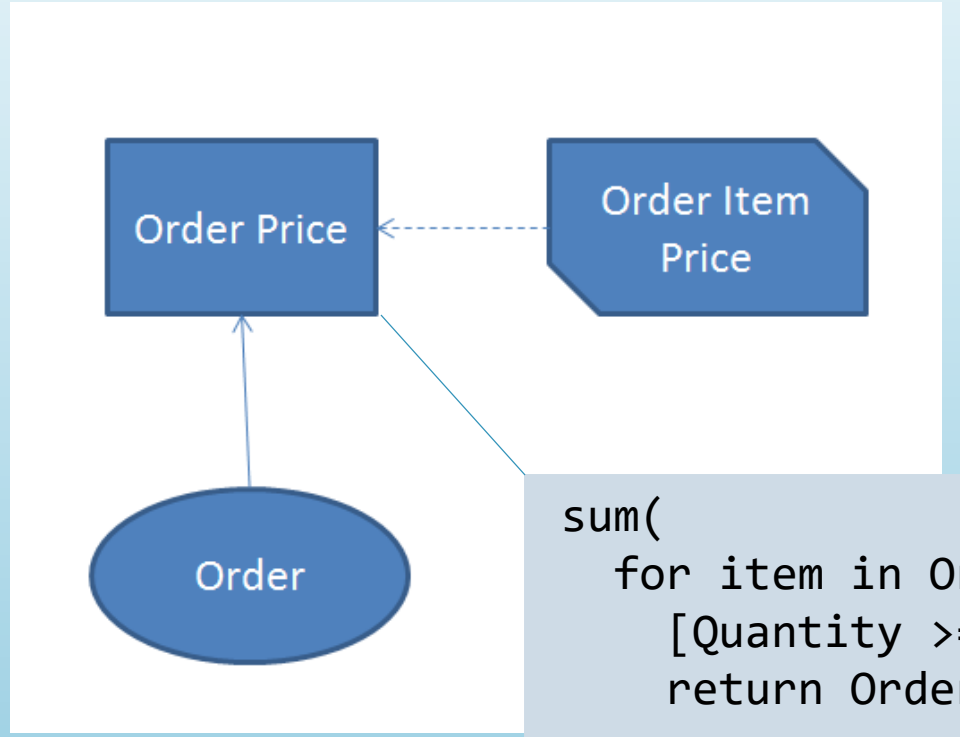
- Members must be Number, Integer, Date, Time, or Duration
- Allow tests such as [0..100] to be passed as parameters or assigned to variables
- Provide Interval Algebra built-ins (union, intersect, meet, *etc.*)
- DMN 1.2 added integer ranges in **for** expressions:

  for i in 1..3 *return i*i*
  returns *[1, 4, 9]*

- We don't have a notation for all positive integers (*>0* denotes all positive numbers)

Iteration – Currently

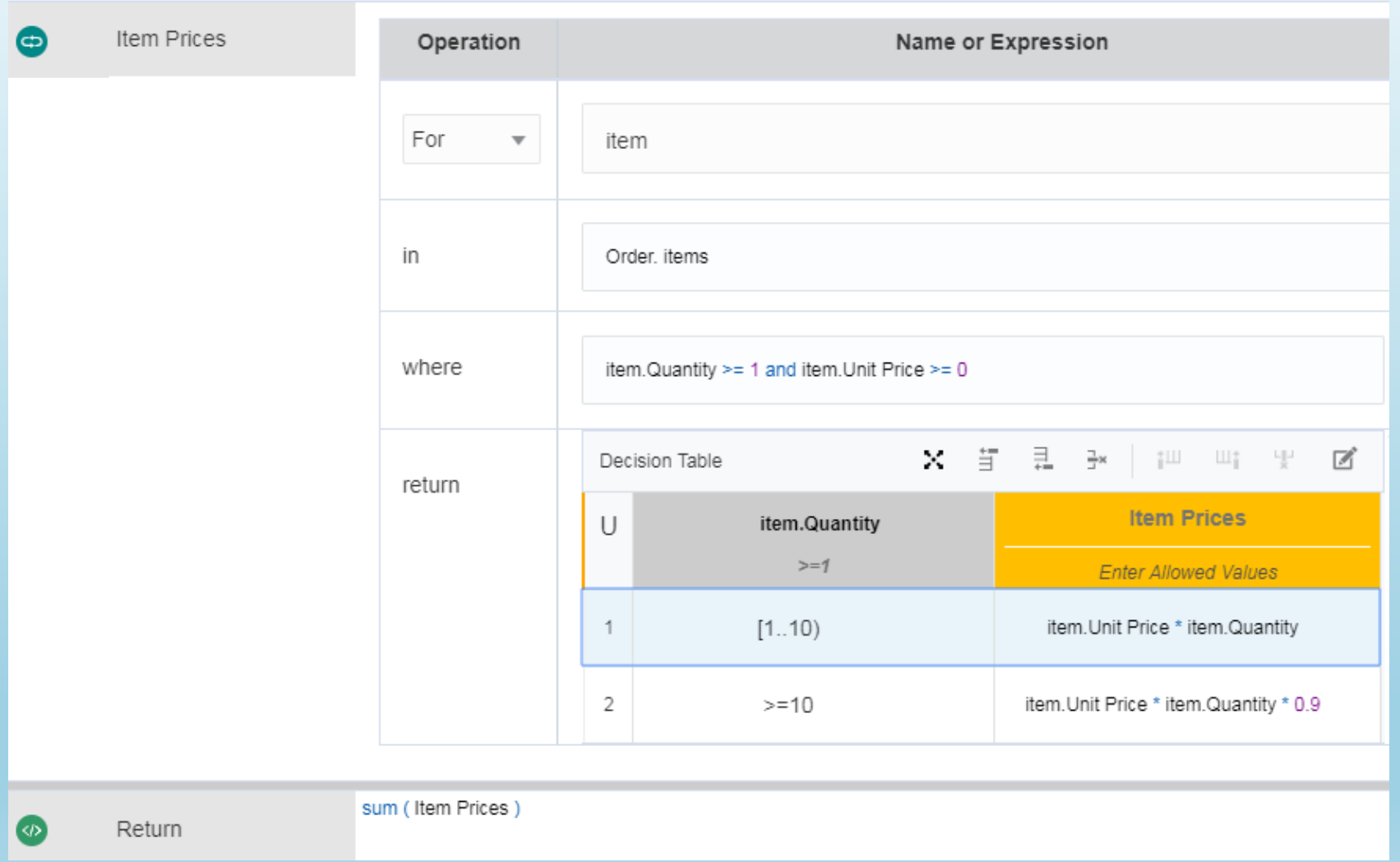
- Use a FEEL **for** loop to apply logic to each element of a list
- If logic is a decision table, it should be a separate BKM




```
sum(
  for item in Order.items
    [Quantity >= 1 and Unit Price > 0]
    return Order Item Price(item))
```

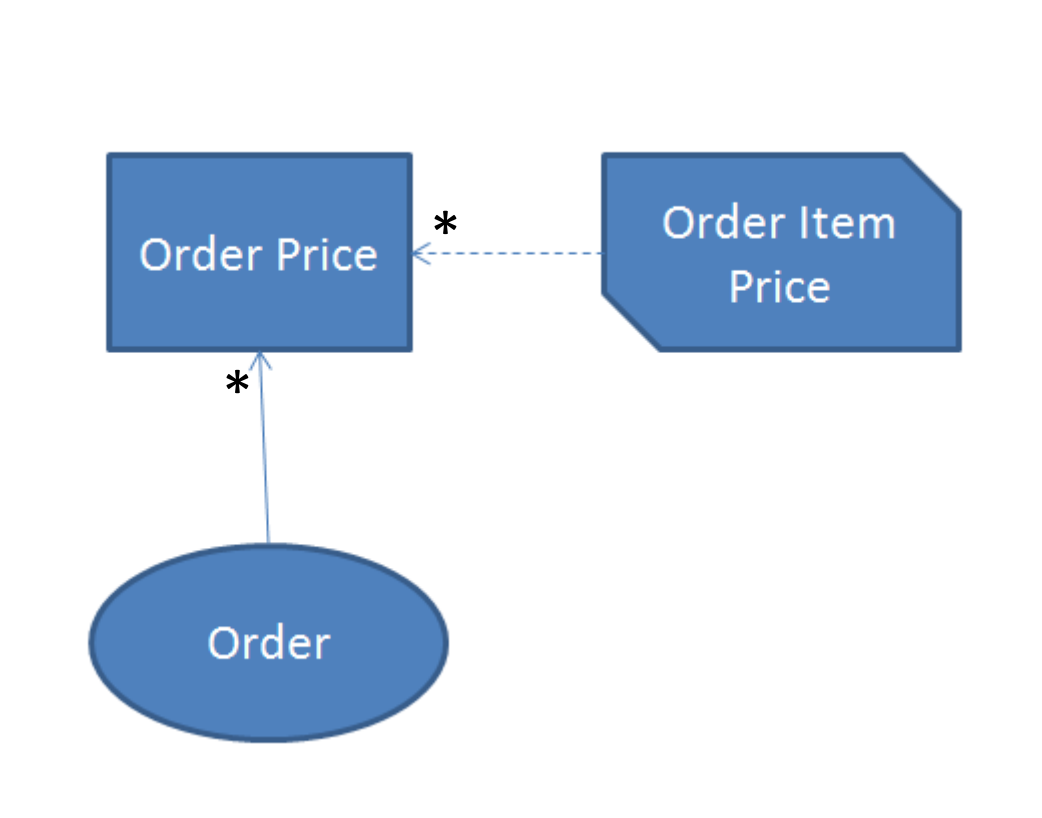
Iteration – New Boxed Expression

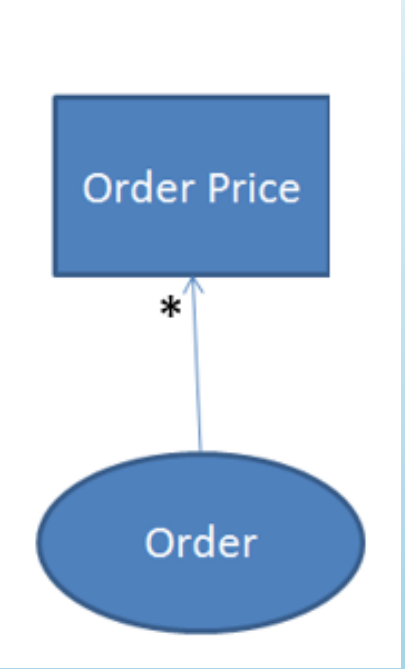
- Use boxed iteration to apply logic to each item of a list
- If logic is a simple decision table, it can be inline
- Extend with XQuery FLWOR clauses such as **where**, **let**, **order by**



Iteration – New Diagram Annotations

- A requirement that is *used* multiple times to produce a single result may be marked with an (*)
- The (*) has no effect on the logic





Recursion

- A Business Knowledge Model, Decision Service, or Named Function can invoke itself directly, or indirectly via another BKM, Decision Service, or Function
- Knowledge Requirements may be cyclic
- A Named Function is a function defined in a context. The Named Function is visible to all context entries.
- Evaluation may not terminate


```
Function(m)
  if count(m. Direct Reports) > 0
  then return sum(
    for d in m. Direct Reports
      return 1 + Total Reports(d))
  else return 0
```

Case Insensitive Names

- To a business person, these names are the same
  - Credit Score
  - Credit score
  - credit score
- And probably these as well:
  - CreditScore
  - credit_score
- Unfortunately, these are all different in FEEL

Unambiguous Context-Free Grammar

- ‘Single Quoted Names’
  - Spaces, keywords, operators can be part of a name, but the name must be enclosed in single quotes
  - ‘Sale amount (USD)’
  - ‘for sale amount’
  - ‘1.0.0.a’
  - Some names illegal, *e.g.,* ‘ ‘
- Typographical notation

“This is some text.”	*This is some text.*
date(“2019-09-16”)	***2019-09-16***
‘This is a name’	<u>This is a name</u>
Different typography should be used for character sets where italics, boldface or underlining are difficult to read or not supported. This is often true of Asian character sets.	

Sequences instead of Lists

Sequences (like Xpath)

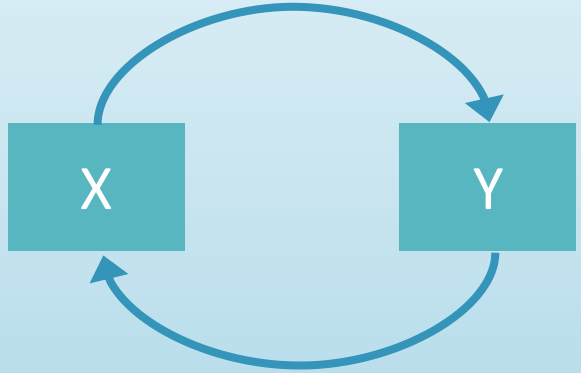
- No nesting
  (1, (2, 3)) = (1, 2, 3)
- A singleton is equal to its element
  (1) = 1
- Comparison
  S1 < S2 = some s1 in S1, s2 in S2 satisfies s1 < s2
- Nested relations are supported
  ({id: 1, children: ({id: 11}, {id: 12})}, {id: 2, children: ({id: 21}, {id: 22})})

Lists

- Can be nested to form trees and matrices
  [[1, 2], [3, 4]] [2][2] = 4
- Singleton is **not** equal to its single element
  We tried that in DMN 1.0, it didn't work out
- *L1 < L2 = null*
- Singletons are common so some attempts are made to implicitly add the filter [1] to avoid clutter and avoid *null* results, but it complicates the spec

Cyclic Requirements, Constraint Solving

- DMN is not powerful enough to solve systems of equations or inequalities
- DMN design assumes an acyclic decomposition of information requirements



$$X = 2Y - 1$$
$$Y = X - 1$$

$$Y = 2$$
$$X = 3$$

Discussion

- Many of the top requested features could be done in 1.x
- We could draft an RFP for DMN 2.0 such that an acceptable submission would be DMN 1.3 plus a few grammar rule changes
- Many features have been in the Revision Task Force backlog for years. There is a lack of volunteers to work out detailed proposals (in part due to the need to be an OMG member)
- These are incremental improvements – is there anything big, new, or disruptive to consider?