Current the Decision Requirements Diagram does not show when data provided by a Input Data or decision is iterated through or aggregated by the dependent decision. It does not depict changes in **data cardinality** or *fan*.

The least impactful way to document fan in a decision requirements diagram is to indicate visually, using a set of three vertical lines[1], which end or ends of an information requirement are multi-instance:

- Where fan-in occurs, indicate that there are multiple instances of the decision producing data which is fed into a single instance of the decision that consumes (and aggregates) it
- Where fan-out occurs because a data input or sub-decision creates a collection, indicate visually that the consuming decision will be invoked multiple times: once per item in the collection.

Figure 1 Decision Requirement Diagram Depicting Multi-Instance Decisions

Figure 1 includes this idea. Multiple *Items* are each processed by their own instance of the *Discounted Item Price* decision (fan neutral). This produces a collection of item prices which fans-in to a single instance of the *Cart Price* decision. The latter decision table definition holds the detail of how the fan-in works (i.e., how the prices are aggregated into a total discount per cart). Similarly, Figure 2 explicitly depicts that for a single customer there will be a collection of *Eligible Products* and each of these will need its own instance of *Cost Product*. In other words the latter decision must iterate over a collection of products.

[1] This symbol is used to be consistent with BPMN which uses it to denote a multi-instance process. We use it to denote the cardinality of a requirements relationship.

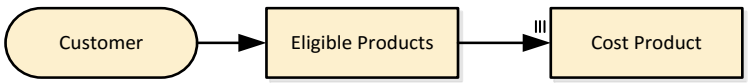

Figure 2 Decision Requirements Diagram Depicting Explicit Fan-Out

Where necessary, a decision table's decision cardinality involves nominating a data class. When a decision table nominates a cardinality it is separately invoked for each available instance of the specified data class. In the case of Figure 2, the decision cardinality is *Product*—therefore the table is invoked once per *Product*.

An explicit statement of a cardinality is not required if it can be unambiguously inferred from input data. If, as is the case in the majority of situations, a decision has only a single data input or multiple data inputs bound by a 1:1 relationship with no collections the cardinality does not have to be explicitly stated.

If there is any chance of ambiguity, the cardinality should be documented in the context of the decision table consuming the data.

Notes

1. Notice that multi-instance symbols are not placed *inside* the decision as this can cause confusion if a single decision that produces a collection provides an information requirement to two consumer decisions—one of which exhibits fan-in (consumer aggregates) and the other fan-out (the consumer iterates). A similar argument exists for decisions with information requirements on data inputs—some may be consumed individually, others iterated and others aggregated. Cardinality is a property of the information relationship between one component and another—rather than a set property of any component. Rather like cardinality of a class relationship in UML is seen as a property of the relationship not either class.
2. Although fan-in suggests aggregation, it is best to leave the mechanism of that aggregation to the Decision Logic Level.
3. The convention (shown in Figure 2) that any decision that produces a collection uses a plural name also clarifies matters.