ISSUE:

Currently the 'choice' construct is restricted to only handling interactions in a single direction, and associated with a single relationship. This is to avoid the 'distributed choice' situation, where each participant (assuming just two participants for now) initiate their own interaction, and these interactions cross - causing both participants to take different paths through the choreography.

However, the cancel pattern (i.e. where a user places an order, but then has the ability to cancel the order providing that the supplier has not sent out the orderCompleted), requires this 'distributed choice' to avoid a very complex (maybe impossible) choreography.

In the current CDL, it would be necessary to define state variables, use the state alignment mechanism and various guards to attempt to ensure each participant takes the same path. This would cause what is perceived to be a simple business pattern to be described in a very complex way - and currently no one has been able to define such an example in CDL (see Steve's "CDL Challenge" email from 23rd June).

PROPOSAL:

The problem with the distributed choice situation is that we need to be able to (1) detect that an inconsistency has occurred, and then (2) have a prior agreement to understand which path has been taken.

Generally when this cross over of interactions occurs, we just want to ignore one of the communications - and therefore we are assigning a priority to one of the interactions. If the priority communication occurs, in a situation where the interactions have crossed, then both parties know that the priority interaction should be obeyed, and the other interaction ignored.

To ensure that this scheme is implementable, we also need to perform verification to ensure that the participant who has issued the non-priority interaction receives confirmation that it has been processed, instead of receiving a higher priority communication. This confirmation can simply be in the form of the next relevant interaction in sequence, following receipt of the non-priority message. Until a confirmation of which path the other participant has taken is received (i.e. a cancel confirm or order completed), the 'user' participant cannot perform any other interactions, as the appropriate path has not been confirmed. This is not the case for the participant that issues the priority interaction, as it will already know which path has been confirmed, and can therefore proceed immediately.

The approach is that any interactions (i.e. there may be more than two elements in a choice statement), other than the highest priority interaction, should await an acknowledgement/confirmation interaction from the other participant before performing any other activities.

For the purpose of clarity, the prioritization of interactions within the choice will be based upon their lexical order. Therefore no changes will be required to the CDL notation - only a description of the semantics associated with interactions that cross over is required and a clear statement of the prioritization based on lexical order.

This approach is no different to how cancellations occur in real business, where the user would either be given a cancel confirmation, or an indication that the order completed (i.e. the cancellation was sent too late), so that they know the state of the business transaction. This way, both the user and the service understand which business flow they have taken.

To illustrate this example using pseudo CDL,

```
<interaction from="buyer" to="seller" operation="placeOrder" />

<choice>
        <sequence>
                <interaction from="seller" to="buyer" operation="orderConfirmed" />
        </sequence>

        <sequence>
                <interaction from="buyer" to="seller" operation="cancel" />
                <interaction from="seller" to="buyer" operaton="cancelConfirmed" />
        </sequence>
</choice>
```

This example represents the CDL for the business flow for handling either an order confirmation or cancellation. If we now add the assumption that the choice elements are in priority order, it then also means that both participants would understand how they should handle the situation if the 'cancel' and 'orderConfirmed' interactions cross over.

The benefit of this approach is that the CDL does not have to explicitly encode state information and decisions required to cope with this situation. The CDL simply outlines the business interactions - which is how a business user would expect to see such a simple pattern described.