# *CDL and Coordination (Business Transactions)*

Tony Fletcher, Peter Furniss, Bob Haugen
27 September 2004
Choreology Ltd

Transactions in a protocol are essentially a mechanism for ensuring state alignment.

In CDL, an interaction can be defined as being aligned (align=true). The alignment attribute is effectively a requirement on the binding that it must, by some means, ensure that the alignment is achieved. Interaction alignment necessarily proceeds step by step. In other words, interaction alignment can assure that parties agree that a particular message with particular content has been sent and received, but cannot assure agreement on propositions that require more than one step.

There is a need for a larger unit of coordination – a set of interactions that end with shared knowledge among the parties that their business relationship is in a defined state: for example, contract agreement. Again, this will be a requirement stated in the CDL that the binding will have to ensure. Such a unit need not be aligned at each step (though it could be) – it is only required that a clear alignment point is made.

The starting point, or basic coordination unit, for a transactional choreography is thus a set of interactions that change the state of the roles such that the changes are subject to the transaction decision – in other words, when the transaction has finally completed, the states are semantically aligned.

It would be possible to identify the interactions making up the coordination unit explicitly with a new construct, but it seems better to use the <choreography /> element. A coordinated choreography can then be used as an element in composition.

## Coordinating a single-level choreography

A one-level choreography (i.e. a root, with no enclosed choreography or performed choreography, and as a root, incapable of being composed in CDL) can be coordinated. In this case the requirement is that the underlying binding ensures that all the roles agree on how the choreography ended – in particular whether it succeeded or suffered an exception. No additional syntax is needed for this, other than an assertion that the choreography is coordinated. The exception block can be used to assert what alignment is achieved in the case of failure (e.g. the exception block could assign values to variables to indicate an order has not been placed). The implementation and binding of such a choreography to real protocols will typically require coordination protocol support to ensure that the parties achieve the consistent result required by the choreography definition.)

**Proposal 1: Add a boolean attribute to choreography: “coordination”**

Since a coordinated choreography may be unsuccessful (i.e. the exception block fires, and all parties know it fired) due to application-defined circumstances (e.g. no stock or bad credit), there needs to be a way to raise an application exception. The CDL spec suggests as much in the text on the exception block, but doesn't seem to specify a mechanism to trigger it.

**Proposal 2: Add a <throw/> or <raise/> activity which causes the exception block to fire.** This will need to have an attribute or element to allow a particular value or exception type to be thrown.

## Coordinating inner choreographies

In addition to one-level choreographies, there are also application cases (such as the TWIST example) where there are inner choreographies that not only need to be treated as a single coordination unit from the perspective of failure, but need to be finalized in a particular way, to ensure business state consistency over the encompassing choreography as a whole. This will often be when some other part of the encompassing choreography fails, and a completed choreography needs to be reversed in some sense. However, a choreography that can be reversed is not really completed – it is an intermediate, provisional state, waiting its finalization. In addition, there are many cases (the TWIST credit check and price request are examples) where the application needs to achieve a temporary provisional ("promised") state from an inner choreography, and then finalize it in one way or another. Different inner choreographies may need to be finalized in different directions in the same instance (as in the TWIST case, where one seller is chosen among those offered).

(Finalizing is not possible with a top-level choreography only because there is nowhere for the finalization stimulus to come from.)

Thus there are two stages to a coordinated non-root choreography:

a) the first, normal (provisional) work of the choreography in the Activity-Notation forming the main body. This activity moves the roles to states where the required final alignment can be achieved by any of the
b) one or more finalizers. Exactly one of these will be triggered by the enclosing choreography, subsequent to the (successful) end of the normal work of this enclosed, coordinated choreography.

If the coordinated choreography is unsuccessful (its exception block fires), the finalizers will not be used, and the exception block should define the state alignment, as for a top-level choreography. As mentioned, if the choreography is successful, the states between the successful end of the normal work and the finalization, although aligned to the extent that they are all awaiting finalization, may not be fully aligned in the way they will be after finalization – for example, business state alignment (e.g. we have/do not have a contract) may only be achieved after the finalizer has completed.

There needs to be a means of explicitly firing finalizers in a "completed" choreography from the enclosing choreography. This is needs to be possible from the normal path of the enclosing choreography, from the exception block and from a finalizer of the enclosing choreography.

The changes needed to CDL to achieve this are:

**Proposal 3: the finalizer already present in CDL should be expanded to allow multiple finalizers, distinguished by an attribute ("case"). Finalizers can only be present in non-top-level, coordinated choreographies.**

**Proposal 4: add a <finalize /> activity, with attributes to identify which choreography instance is the target and which of its finalizers (i.e. which "case") is intended.**

**Proposal 5: add an attribute to <perform /> to assign an identifier to an instance of use of an inner choreography so that it can be referenced by a subsequent <finalize/>**

The "case" values are defined by the designer of the coordinated choreography. It is likely that many choreographies will just have two values – confirm/cancel, accept/decline, drawdown/replenish – but others may have more – buy/sell/cancel. It is not really possible to have only one – the looming possibility of the finalizer must somehow be removed, which implies at least an empty alternative finalizer to be fired (empty in the sense that there is no behaviour visible in the choreography, other than the de-installation of all the finalizers).

## Composing or "overlaying" coordination

As mentioned above, the use of the coordinated attribute on a choreography will typically imply that a coordination protocol is needed to ensure the parties receive the appropriate signals. However, many applications will need to intermingle application messages with the coordination signals (again, the TWIST example – especially on the Buyer:TradingSystem exchange, and possibly on all of the relationships). This could be done by requiring the application protocol specification to include appropriate interactions, including the exception paths and collision resolution cases. The proposals above can be used this way, helping to structure the CDL but not implying any particular underlying protocol mechanism.

However, there is considerable benefit to be gained if it is possible to compose choreographies by "overlaying" them, such that the roles and interactions of one (coordination-oriented) choreography have an explicit equivalence and mapping to the roles and interactions of another, application-specific choreography. The attached xml gives an example in outline of this. This would then make it possible for a binding of the coordination-oriented choreography to a particular coordination protocol to be "inherited" by the application protocol. There will typically still be a need for some

binding information – (e.g. there is the question of which protocol is carried in the headers, which in the body for the different interactions of the lifecycle). The application business protocol designer can then concentrate on the business problem, knowing that the underlying coordination mechanisms will make sure that the appropriate signals will be delivered – all that is necessary is to define what happens when they arrive, and what application messages, if any, are to be sent with them.

**Proposal 6: Add to the composition mechanism (perform) the ability to "overlay" choreographies, such that the interactions of one are *ipso facto* interactions of the other.**

If this is done with <perform/> it can be expressed by allowing the equivalent of <bind> for roles (making a Participant type in the enclosing choreography) and for interactions. Alternatively it might be more elegant to express it as some kind of inheritance, so the application choreography is defined as an extension of the coordination choreography.

It is a requirement that overlayed choreographies are congruent, such that mapped roles share relationships (so if buyer maps to coordinator and seller to participant, then there must be buyer-seller AND coordinator-participant relationships in the application and coordination choreographies respectively).

Note that there could be more than one coordination choreography with different characteristics. Some coordination choreographies could be supported by multiple alternative coordination protocols; others might differ precisely to make use of a facility available in a particular protocol.

Note that the features required to overlay choreographies also make it straightforward to interleave choreographies (as partial business protocols) between the same or different entities.

Overlay choreographies should also enable separation of responsibilities in multi-party transactions like TWIST, where the sellers and credit services should not know about each other and do not need to know about the whole choreography. The choreographies in which they are involved can be published to them separately, and then composed and coordinated into the whole scenario by the Trading Service.

**Example overlaid choreographies**

The example choreographies are based on the CreditCheck choreography from the TWIST example, using Steve's August draft of the TWIST case as the starting point (and partial exemplar) Only limited attempts have been made to get the syntax right – in particular most of the type declarations have been omitted (though the xml is well-formed).

There are four choreographies shown:

Coordination – the representation of the generic coordination protocol. This is a generic choreography on a generic protocol – the propagation interaction does not detail the underlying register/enrol message exchange, for example. In this form, the collision resolution is assumed to dealt with by the coordination protocol, and does not appear in the choreography.

CreditCheck – this is the application specific part. It includes the statement of what happens in application terms when the various events occur. As written, it shows messages being sent at every point – if there is no application information on the drawdown and replenish messages, the finalizers can be simplified to a simple assignment to the state capture variable that summarises the state.

CoordinatedCreditCheck – this is the (incomplete) "overlay" choreography which specifies that CreditCheck uses Coordination. The binding of operations will specify, for example, that CreditCheck's creditAuthorized message maps to Coordination's prepared.

Enclosing – this just shows some fragments of the code that will be in the outer choreography. The CoordinatedCreditCheck is run for seller A and separately for seller B (this will involve another set of <bind>s, of course), establishing (if successful) a credit reservation. Later, depending on other bits of choreography logic, it is determined which of A or B is needed (if either), and the appropriate finalizers invoked.

Note: the generic Coordination choreography can be overlaid on every transaction in the TWIST scenario: between Buyer and Trading Service, between Trading Service and Sellers, and between Sellers and Trading Service and Credit agencies.