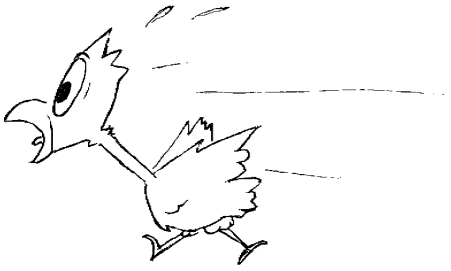


Customer to Distributor: I need 1000 fresh frying chickens for a big banquet, delivered by 4 pm tomorrow. If you can get all 1000 delivered by that time, we have a deal. If you can't, forget it. That's my **drop-dead** date.

Distributor: I'll see what I can do and get back to you in half an hour.

Distributor to Carrier: When would you need to pick up 1000 frying chickens from my Supplier's dock in order to deliver them to my Customer's dock by 4 pm tomorrow?

Carrier: To be certain, 1 pm tomorrow. Do you want to schedule the pick up?

Distributor to Supplier: Can you get 1000 fresh frying chickens ready for pickup by 12:30 pm tomorrow?

Supplier: Yes I can, do you want me to do it?

Distributor to Customer: I can deliver the chickens by 4 pm tomorrow.

Customer: It's a deal.

Distributor to Supplier: Yes, get the chickens ready for pickup.

Supplier: It's a deal.

Distributor to Carrier: Yes, pick up the chickens and deliver them.

Carrier: It's a deal.

Consider hundreds or even thousands of orders of equal or greater complexity per day. (Don't get hung up on the chickens, they're only an example.)

*Now you know why people want to do **multi-party electronic business transactions**.*

Chicken cartoon by Tom Haugen

Multi-Party Electronic Business Transactions

Version 1.1
7 December, 2002

Authors

Bob Haugen, Logistical Software LLC, ebXML and UN/CEFACT TMG Business Process Work Group
Tony Fletcher, Choreology Ltd, OASIS Business Transaction Protocol Committee and UN/CEFACT TMG Business Process and Architecture Work Groups

Abstract

We present a business scenario, the Drop-Dead Order, that is best handled as a multi-party electronic business transaction. We present models of such transactions using the UN/CEFACT Modeling Methodology (UMM) and the OASIS Business Transaction Protocol (BTP). We claim that the two models are sufficiently equivalent that the same runtime software could execute either. We recommend that BTP be considered as an underpinning implementation technology for UMM, thus harmonizing two specifications that have been considered incompatible. We suggest that further efforts be made to harmonize the other major proposals for electronic business processes so as to converge from a confusion of incompatibility to one global standard, or at least good interoperability.

Along the way, we dip into several other important topics, such as the benefits of transactional behavior, the meaning and scope of transaction contracts, the use of coordinators and commitments to structure complex business collaborations, and the distinction between business coordination and transaction completion.

.

Business Scenario

Drop-Dead Order

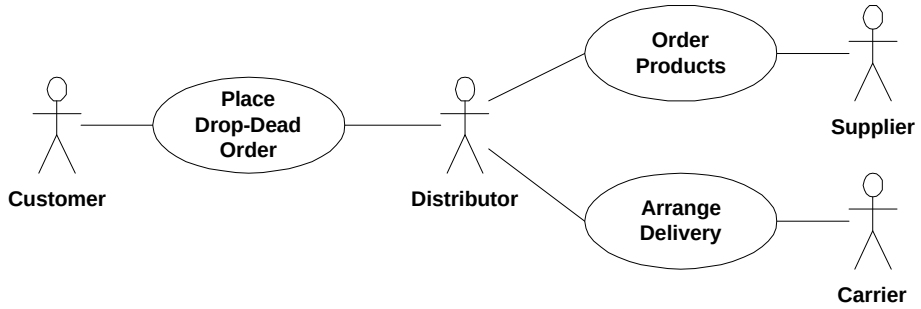


Figure 1: Drop Dead Use Case

The Customer orders products *only if* they can delivered by a hard deadline – the **Drop-Dead Date**.

The Distributor does not have products on hand; needs to order products from a Supplier, plus arrange shipping service with a Carrier (drop ship from Supplier to end Customer).

Both product order and shipping order could use the Supplier Cascade pattern, where the buyer goes through a list of suppliers until success or failure (failure = end of list without success). In fact, the customer order could use the Supplier Cascade pattern, too, if the first Distributor can't deliver.

If Distributor is able to get commitments from *both* product Supplier and Carrier,
    then Distributor accepts Customer order
    and also confirms commitments from Supplier and Carrier.
Otherwise,
    Distributor declines Customer order,
    and also cancels commitments from Supplier and Carrier.

To put it another way, the Distributor wants both the chickens and the delivery service, but if not both, then neither. The Distributor does not want to get stuck paying for chickens that can't be delivered, or an empty truck. You could call this an **All-Or-None** situation.

In this document, we will detail only the order creation stage of the scenario. In real life, of course, the Customer wants the products to be delivered, and the Distributor, Supplier and Carrier want to get paid. In a sequel to this document, we could cover those stages of the business collaboration. Some of them would be multi-party transactions, too.

Definitions of Concepts

The assumed level of prior knowledge for readers of this document climbs from "not much" at the beginning to "student of UMM and BTP" in the last sections. References abound at the end.

"Transaction" is one of the most overloaded words in the electronic business vocabulary.

By "electronic business transaction", we mean "transaction" in several senses of the word:
- business actions that go across (Latin *trans* – across, thus *transaction*) two or more parties;
- an "atomic unit of work" in which many actions may be involved, but they all succeed together; or if any single component action fails, they all fail;
- a consistent change in the state of a business relationship between two or more parties, where each party agrees to the change in state, and knows that the change was consistently implemented by all parties.
    - (This is like the **handshake** upon agreement in face-to-face deals.)

Several groups have proposed protocols for managing electronic business transactions. For example:
- RosettaNet, ebXML, and UN/CEFACT BCP (Business Collaboration Protocol) – all related
- OASIS-BTP (Business Transaction Protocol)
- WS-Transaction (WS-T), one of three related specs with WS-Coordination (WS-C) and Business Process Execution Language for Web Services (BPEL4WS).

The different protocols provide **different sizes or scopes for business transactions**:
- RosettaNet, ebXML and UN/CEFACT BCP promise two-party business transactions.
- ebXML and BCP provide multi-party business collaborations without transactional behavior.
- OASIS-BTP and WS-T promise multi-party (two or more parties) business transactions.

This document intends to take **a small step toward harmonization**, if not convergence, of the above "standards". We will describe a multi-party business transaction scenario that could work under any of the above protocols. We will show how it would work under both UN/CEFACT-BCP and OASIS-BTP. WS-T is left as the proverbial exercise for the reader - or a sequel.

Keys to convergence:
- Break the multi-party scenario into smaller two-party interactions as in RosettaNet, ebXML and BCP.
- Create Coordinators as defined by BTP to manage the interdependencies of the smaller interactions.
- Have the Coordinators use Economic Commitments as defined by UN/CEFACT, to give structure and logical meaning to the interdepencies of the smaller interactions.

Why is this important?

- Transactions give predictable **closure** or **completion** to electronic business dealings, answering these and other important questions:
    - Are we done yet?
    - Did that group of actions succeed or fail?
    - Do we agree on what happened (success or failure)?
    - Can we prove (maybe in court) that the group of actions succeeded or failed?
- Business deals may involve many parties and actions, all of which may need to succeed or fail together (as in atomic transactions).
- A confusing mix of electronic business transaction protocols gets in the way of doing business.
    - The Internet and the World Wide Web enable global electronic communication via global standard networking protocols.
    - Global business transaction standards will similarly help to enable global electronic commerce.

Pairing off

No doubt business collaborations may involve many parties. In international trade, the number of parties may multiply by ten.

But powerful forces will encourage mobs of parties to pair off into sets of couples:

- **Simplicity** – two is *much* simpler to handle than more-than-two.
- **Accountability**
  - Business contracts may involve many parties, but well-formed contracts always specify exactly who has agreed to do exactly what for exactly whom (i.e., only two parties to each contractual commitment).
  - The more parties to a contract, the more parties have to agree on *everything*.
  - Economic commitments and events always involve two and only two parties.
- **Security** – need-to-know. Trading partners need to know about each other, but they rarely need to know what other partners each may have, and in many cases *should not know even that others exist*.
- **Decoupling to avoid ripples of change** – if parties A and B change their procedures, it may not affect parties C and D, unless all four parties are bound into the same contract.

Types of Contracts

Business trades are governed by many different kinds of agreements, some made by the trading partners amongst themselves, some imposed upon them by laws, governments and customary practices.

The Drop-Dead scenario involves two different kinds of legal contracts: Transaction Contracts and Economic Contracts. Economic Contracts are well known. Transaction Contracts seem to be less well-known, but we think they are also very important.

Transaction Contracts are the main focus of this document.

Transaction Contract

We use the term "Transaction Contract" to mean agreements between trading partners about the technical means and logical rules by which they will execute trades with each other. (Note: "technical" does not necessarily mean "computer". On the other hand, in this document, we will focus on transaction contracts that are enforced by computer software.)

You might compare Transaction Contracts to Robert's Rules of Order for meetings, or the rules of a game like Monopoly. A United Nations recommended Transaction Contract, called *Electronic Commerce Agreement*; is listed in the References. It is one of the foundations for the UN/CEFACT Business Collaboration Protocol, one of the protocols used in this document..

Transaction Contracts contain rules about two different (but complementary) sets of interactions:

- **business interactions:** that is, offer-acceptance, commitment-fulfillment, claim-settlement;
- **transaction completion interactions**: that is, success, failure, confirmation, cancellation, etc..; bringing a unit of work to a determinate end.

Business interactions rules are like Robert's Rules for taking turns to speak, or speaking to the issue on the table. They answer questions like:

- What's the process for creating an order?
- What is the format of the business documents that we will exchange?
- What business transaction pattern and protocol will we use?
- At what point can I be sure that we have a deal, so I can act on it?
- If we negotiate an order through several versions, how do we know which one should be fulfilled?
- What is the process for resolving fulfillment discrepancies?
- What are the preconditions for payment?

Transaction completion interaction rules answer more mechanical questions:

- What are the boundaries of this unit of work?
- Are we done with this unit of work yet?
- If something went wrong technically, can we recover? If so, how?
- Did this unit of work succeed or fail?
- Can we prove that it succeeded or failed?
- Can we prove it in court, if necessary?

At this stage of development, transaction contracts may not be expressed in any one place, but may be interleaved through many places. For example,

- Trading Partner Agreements (TPAs),
- ebXML BPSS documents or BTP contracts,
- terms and conditions in Economic Contracts,
- or all of the above.

However, we think it is valuable to factor out the Transaction Contracts from all of the surrounding agreements, and also to factor out the business interactions from the transaction completion interactions. For one reason, the same transaction completion interactions will be used for many different business interaction patterns.

Incidentally, the scope of Transaction Contracts is not necessarily the same as the overall business scenario, especially in a multi-party scenario. Transaction Contracts are only required between trading partners that have direct interactions with each other. In the Drop-Dead scenario, Transaction Contracts exist between Customer and Distributor, Distributor and Carrier, and Distributor and Supplier. The Transaction Contracts were agreed upon first, to enable the making and fulfillment of Economic Contracts (Orders).

Economic Contract

Economic Contracts involve exchanges of economic resources between economic agents. That is, they are about buying and selling goods and services. If money's involved, it's probably an Economic Contract.

Orders like the customer order and the product and shipping orders in the Drop-Dead scenario are Economic Contracts. They could also be governed by long term (e.g. yearly) Economic Contracts.

If the entire multi-party deal is successful, the Economic Contracts will work as follows:

- The Customer will have a contract with the Distributor.
- The Distributor will have a contract with the Supplier.
- The Distributor will have a contract with the Carrier.

The Drop-Dead scenario has no multi-party contracts. They're all in pairs.

Other scenarios might involve multi-party contracts. But they would normally be broken up into two-party Commitments. (See below.)

Economic Commitments

Economic Commitments are promises from one party to perform a specified Economic Event in the future for another party. Usually they involve promises to deliver goods or services, and reciprocal promises to pay for them. Contracts (e.g. Orders) may contain many Commitments (e.g. Line Items).

For example, an Order Line Item often represents two reciprocal commitments:

1. from the seller, to deliver goods or services;
2. from the buyer, to pay the price.

We separate the two Economic Commitments from an order line item in electronic business collaborations because each commitment will be fulfilled by different economic events, e.g., delivering the goods and paying for the goods.

Well-formed Economic Commitments always involve two and only two parties. Otherwise problems of accountability arise:

- Who is supposed to do what for whom?
- Who is supposed to decide if it was done satisfactorily?

With two parties, it's clear. With three or more, disputes are probable.

Offer-Acceptance

The typical business protocol for the formation of contracts, orders, and legal commitments of all kinds is called Offer-Acceptance.

One party makes an Offer (to buy or sell), and the other party either Accepts or Declines the Offer.

The rules for how Offer-Acceptance transactions work are part of the transaction contract, and may be spelled out by by adopting the UN/ECE standard, or in a Trading Party Agreement that covers many Offers, or in terms and conditions of the specific Offer under consideration, or in some other way. But best practices in electronic commerce suggest that they be spelled out and agreed to very explicitly, because the rules have legal significance, especially when something goes wrong.

"When something goes wrong" is also the motivation for transaction protocols – to minimize the opportunities for something to go wrong, and define what happens if it does. The transaction protocol to be followed would be part of the Transaction Contract between trading partners.

Offer-Acceptance Variations

There are many variations of Offer-Acceptance collaboration patterns. We list only a few.

Order-Driven [Binding Offer]

In an Order-Driven scenario, the Requester initiates the business collaboration by making the offer.

If the Offer is a Binding Offer, then if the responding party Accepts it as offered, they have a deal. The only excuse for the offering party to declare failure would be if the Acceptance did not follow the Transaction Contract rules. For example, if the Acceptance was too late, or the Acceptor tried to change the Offer, or the Acceptance was not digitally signed or was signed by an unauthorized party, etc.

Quote-Driven [Binding Quote]

In a Quote-driven scenario, a buyer initiates the business collaboration by requesting a quote.

For example, in financial service industries, a buyer may request a quote for some stock shares. The seller sends the quote (e.g. price per share and quantity available). If the buyer places an order within the conditions of the quote, the seller is obligated to accept and they have a deal. The only excuse for the seller to reject the order would be if the buyer deviated from the conditions of the quote, for example, stated a different price, ordered more than was available, was too late, etc.

Contingent Offer

Trading parties are free to make any Transaction Contract rules they want. The Offer might be contingent, and the offering party might be free to reject an Acceptance. By "contingent", we mean "the offer may be revoked due to circumstances beyond the interaction of the two parties in the offer-acceptance transaction".

We'll use this variation between Distributor and Supplier and Carrier in a Transactional Collaboration model later in this document. The Distributor proposes an offer to Supplier and Carrier that is contingent on the acceptance of both Supplier and Carrier as well as the successful conclusion of the order from Customer to Distributor. This variation appears to abuse some rules of UN/ECE and UMM, as we will explain in the details of the model. We show some compliant variations in this document as well as a separate Appendix.

Negotiation

See the ebXML Negotiation Pattern in the References.

Notes on the UMM Models in this document

If you are a UMM expert, please read this section carefully.

Methodology vs Metamodel

UMM is a methodology, which also incorporates a metamodel specifying logical elements and relationships for business collaborations (the Business Collaboration Framework and Business Collaboration Protocol).

We are not strict about our use of the UMM methodology in this document, but we are trying to be true to the spirit (and we also think 94.78% of the letter) of the UMM metamodel.

Divergences from UMM:

- We will not use the whole methodology, but will only show selections to illustrate how to use the UMM Business Collaboration Framework and BCP to implement a multi-party transaction.
- We will cheat a little on some of the diagrams – mixing some elements from different diagrams to make the story clear. *We do not intend these mix-ins as changes to UMM.* In the Activity graphs, the mix-ins are shown in color.
- As noted above and below, the Contingent Offer transactions between Distributor and Supplier and Carrier abuse the UMM receiptAcknowledgment signal. We will present some models that follow UMM to the letter in this document in an Appendix, but we chose to show the Contingent Offer because it provides the cleanest transaction completion.

Multi-party Transactions

UMM Business Collaborations may enclose and coordinate many UMM Business Transactions. UMM Business Transactions, as we have said, always involve two and only two parties, while UMM Business Collaborations may involve many parties.

However, UMM Chapter 8 explicitly states:

> 'A business collaboration protocol is not a transaction and should be used in cases where transaction rollback is inappropriate.'

That statement could be interpreted in various ways:

a) as policy, that is, transactional behavior at the Business Collaboration layer is a bad idea; or
b) as disclaimer, that is, UMM does not provide ready-made features or guarantees for transactional behavior at the Business Collaboration layer.

In this document, we will demonstrate that it is possible to implement transactional behavior at the UMM Business Collaboration layer, over many parties and transactions. Therefore, the appropriate interpretation of the UMM statement is b) "no guarantees", not a) "you can't do that".

Moreover, we recommend that the above statement in UMM be changed, for example to read:

> 'A business collaboration protocol does not, of itself, provide transactional guarantees. However, it is possible to implement transactional behaviour in business collaborations by careful modelling and coordination, or by 'wrapping' it with a transaction management protocol, which also provides this coordination (as shown in this paper). A UMM Business Collaboration consists of one or more Business Transactions. The outcomes of the business transactions that compose a business collaboration may be coordinated such that part, or all, of it becomes a transaction in a logical sense.'

UMM Multi-Party *Non-Transactional* Models

The Drop-Dead scenario may be modeled as a UMM Business Collaboration in several ways:

- **Model 1: Separate the Offer of the Customer Order and its Acceptance into two different BusinessTransactions**. Try to get commitments from the Supplier and Carrier in other transactions in between the Offer and Acceptance. Accept the Customer Order if you get commitments from both Supplier and Carrier that fulfill the Customer's specifications, otherwise decline.
  - However, this model requires canceling one of the Supplier or Carrier commitments if the other fails.
  - You could also get temporary holds or self-canceling commitments from Supplier and Carrier, and thus not need to cancel in case of failure. (This is the pessimistic style.) But then you would need separate confirmation transactions in case of success.
- **Model 2: Nest a BusinessCollaboration as a subactivity inside the RespondingBusinessActivity of the Customer Order transaction** and try to get commitments from Supplier and Carrier in the nested subactivity.
  - The same needs arise for separate canceling or confirming transactions, depending on pessimistic or optimistic expectations.

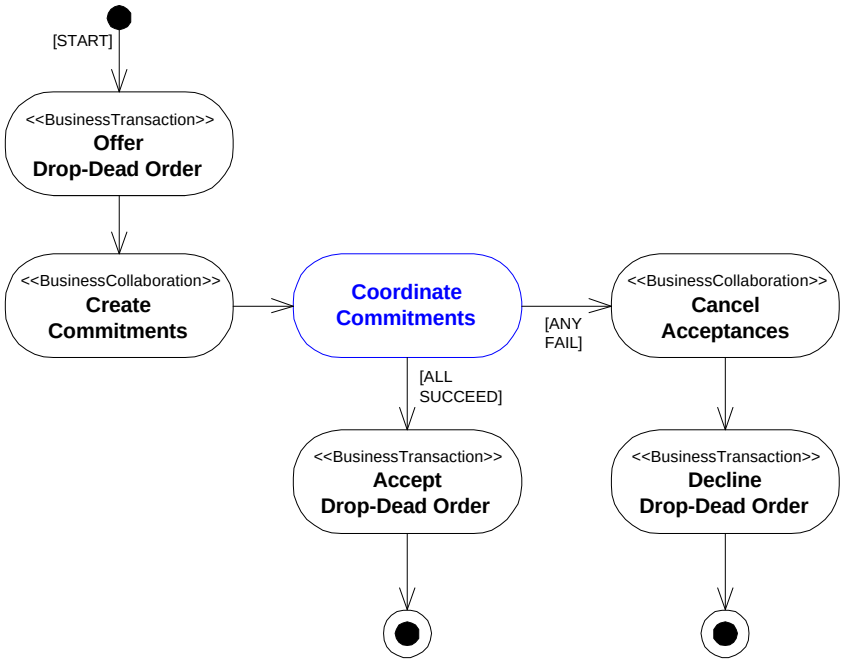


Figure 2: Model 1: Separate Offer and Acceptance

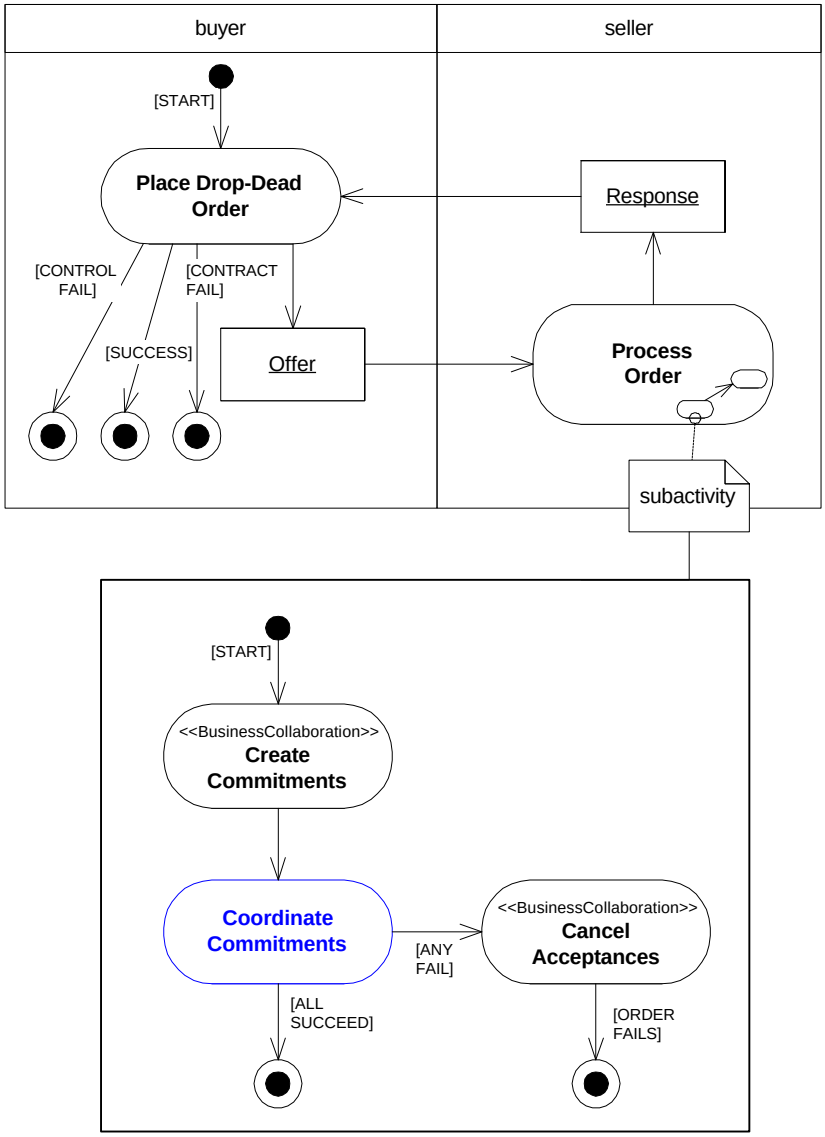


Figure 3: Model 2: Nested Subactivity

Problems with separate canceling or confirming transactions might include:

- Order cancellation not allowed (for example, it's a rush order and the chickens are already "committed").
- Temporary or conditional commitments not provided.
- Order cancellation or confirmation transaction could fail.

In other words, the non-transactional models provide several **opportunities for something to go wrong**:

1. The Offer transaction between Customer and Distributor.
2. The Cancellation or Confirmation transactions between Distributor and Supplier or Carrier.
3. The Acceptance or Rejection transaction between Distributor and Customer.
4. The Confirmation transactions between Distributor and Supplier and Carrier must be completed successfully before the Acceptance between Distributor and Customer. But if the Acceptance transaction fails, another set of Cancellation transactions will be required between Distributor and Supplier and Carrier.

A business process with no need for separate canceling or confirming transactions – where every related offer-acceptance transaction succeeded or failed together – would avoid many of those problems.

How to evolve to a multi-party transaction

The non-transactional models described above contain a clue for a multi-party transaction model. Both of the above models would use something like BTP Coordination logic to decide whether the Customer Order could be fulfilled or not, depending on the results of commitments from Supplier and Carrier.

If we **put the Coordinator in a *different position* in the Business Collaboration structure – as a peer to the Requesting or RespondingActivities in the BusinessTransactions, intervening before each transaction is complete –** then we can get the desired transactional behavior across all of the related Business Transactions in a multi-party Business Collaboration.

Model 3: UMM "Transactional Collaboration" Model

This model breaks everything down to two-party transactions, because that's the basic unit of work in UMM. But it's also the same transaction *contract* scope in BTP, and also the same scope as an Economic Commitment in business contracts.

This model then adds a BTP-style Coordinator to implement transactional behavior over all of the related two-party transactions, in other words, implementing a multi-party transaction. *BTP-compliant software could manage UMM Business Collaborations using this model (or any of the other models we present), with little or no change in either the model or the software.*

The Distributor is the party with Commitments to more than one other trading partner. So the Distributor must coordinate the Commitments with the product Supplier and the Carrier. We'll call Coordinator activity "CoordinateCommitments". It coordinates the commitments from the Supplier and Carrier to determine if they fulfill the order requirements from the Customer as to product specifications, quantity, time constraint, etc. *We will explain the components of this complex diagram in the next few pages.*

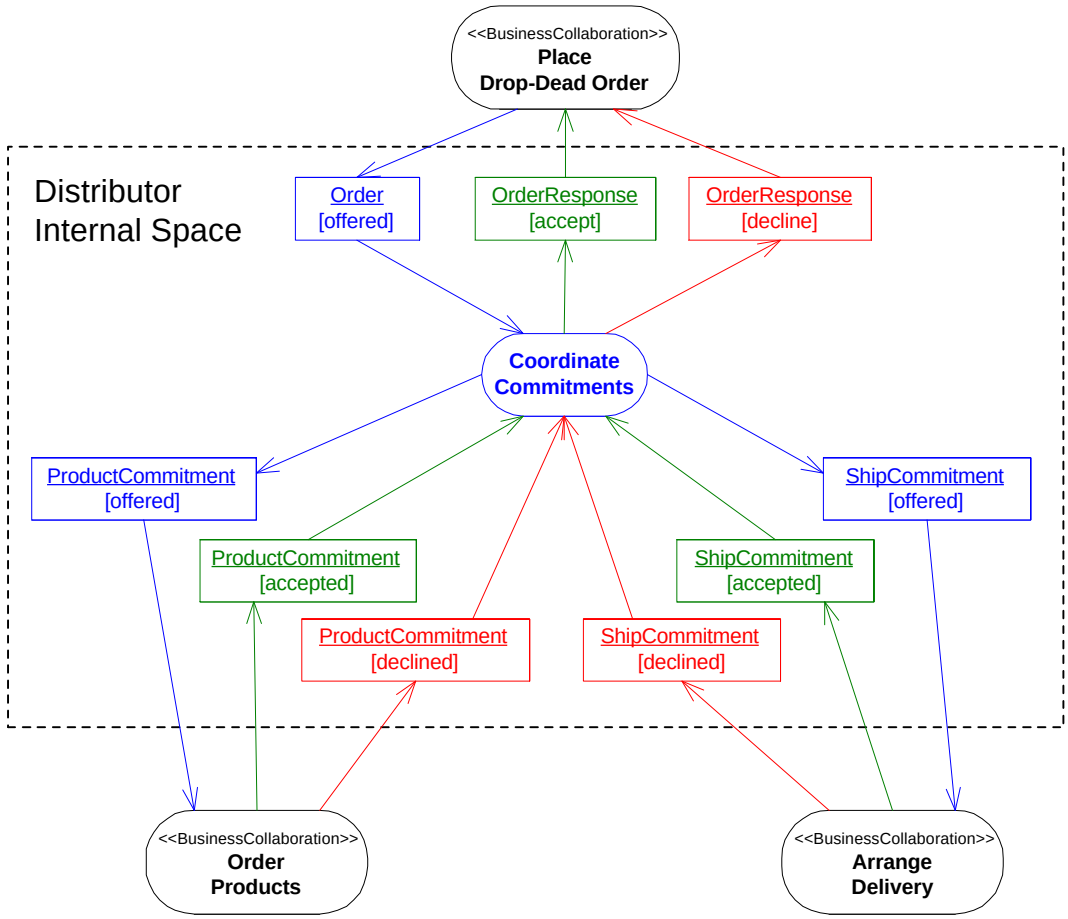


Figure 4: Coordinator Space

UMM (like most ecommerce models) differentiates between logical spaces that are **internal** (private) and **external** (shared between trading partners). CoordinateCommitments is an **internal** activity of the Distributor. **It does not need to be known to Customer, Supplier or Carrier.**

Where do transaction Coordinators fit into UMM?

UMM consists of several Views:

- Business Reference View
- Business Requirements View
- Business Transaction View
- Business Service View

Coordinators fit into the Business Transaction View.

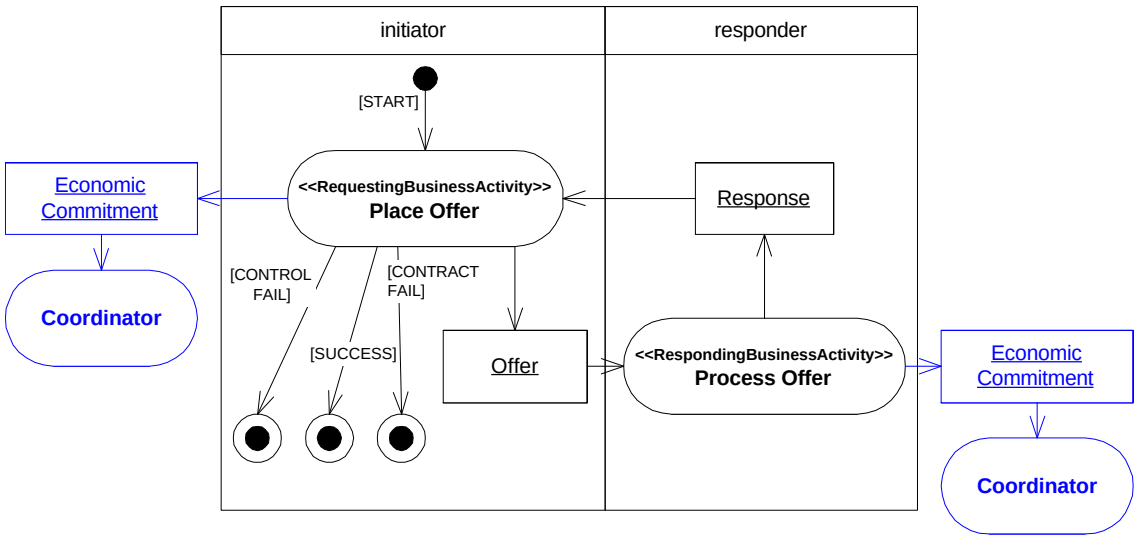


Figure 5: Coordinator in Business Transaction View

These Coordinators interact with Requesting and Responding Business Activities in Business Transactions, to make decisions involving more than one transaction *before each transaction is completed*.

EconomicCommitments are business objects that are instantiated based on business documents (Offer and Response in the above diagram). Coordinators would be used by parties who have dependent commitments from more than one other party.

In the Drop-Dead scenario, the Distributor tries to get commitments from three parties, and the commitments are all dependent on each other:

- A commitment from the Customer to pay for the chickens if the Distributor can deliver them by the Drop-Dead Date.
- A commitment from the Carrier to deliver the chickens by the Drop-Dead Date;
- A commitment from the Supplier to make the chickens ready for delivery.

The Coordinator may interact with many Business Transaction Activities, collecting EconomicCommitments from different parties, and making decisions on transaction responses based on the collected commitments.

As we saw above, Coordinators could interact with either Requesting or Responding Business Activities in UMM Business Transactions. In this Drop-Dead model, we use both variations:

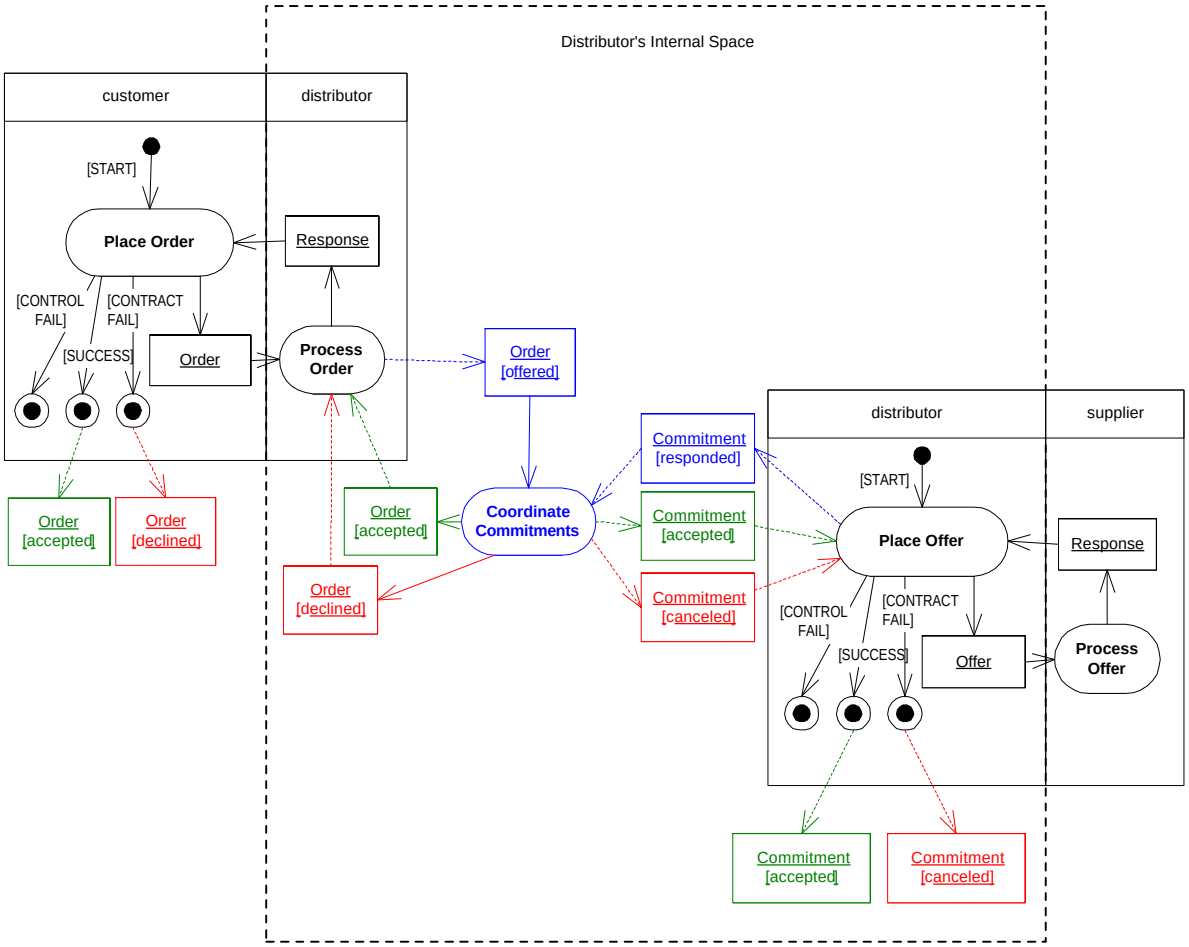


Figure 6: Coordination between transactions

The above diagram shows the transaction between Customer and Distributor on the left, coordinated with the transaction between Distributor and Supplier on the right. If this page had enough space, another transaction could be shown on the right, between Distributor and Carrier, linked to the same Coordinate Commitments activity.

For the transactions between Distributor and Supplier and Carrier, the CoordinateCommitments activity will interact with the RequestingBusinessActivities.

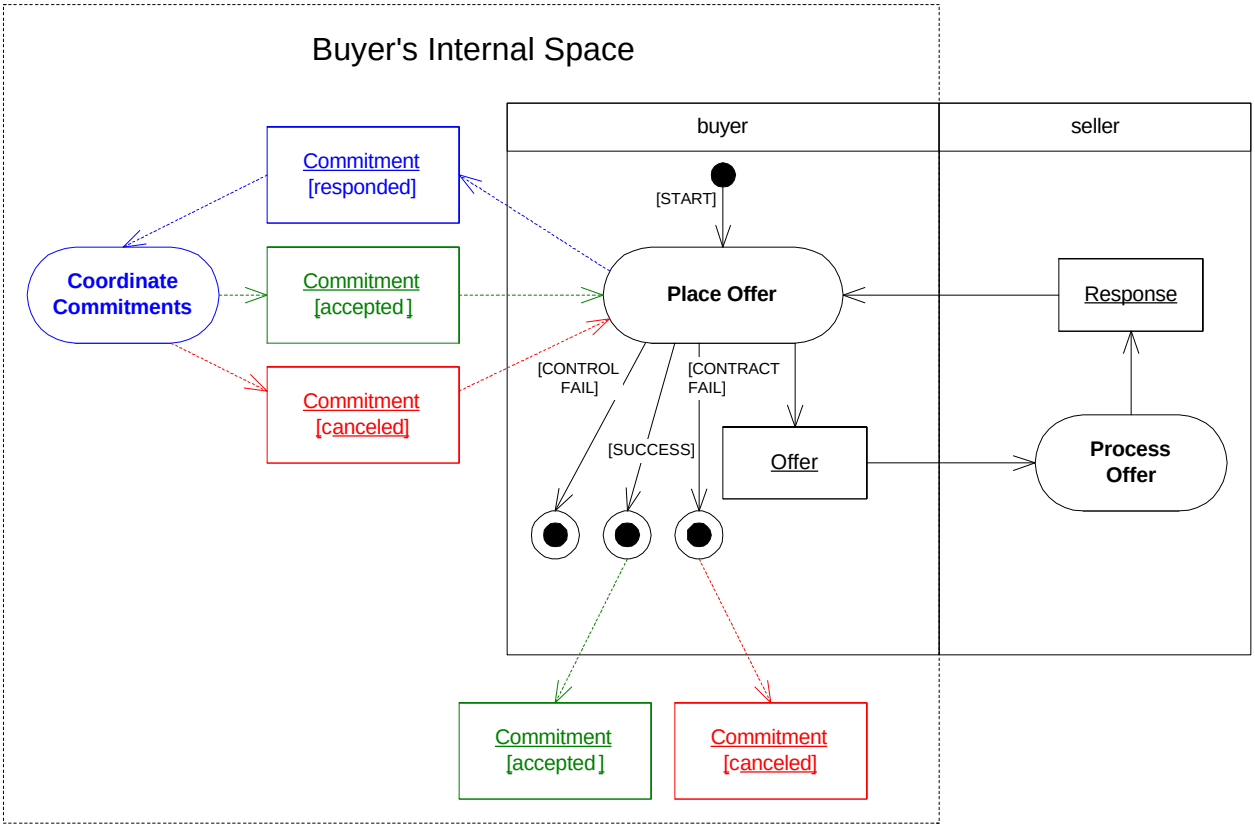


Figure 7: Coordination between Distributor and Supplier and Carrier

This means that:

- the Distributor makes a *non-binding* or contingent offer,
- the Supplier or Carrier responds to say whether they can commit or not,
- and then the Distributor makes a final decision (after getting responses from both Carrier and Supplier, responding to the Customer, and getting an acknowledgment back from the Customer) with a positive or negative acknowledgment of the responses from Carrier and Supplier.
    - The acknowledgments are not shown on the diagram above, but they are shown on the next (Message Sequence) diagram.
    - *Note: UMM does not intend using the final receipt acknowledgment in a transaction to cause a transaction to fail for contingencies, that is, for reasons beyond the interaction between Requester and Responder. We will present in an Appendix a model of interactions between Distributor, Supplier and Carrier that does not abuse the receipt acknowledgment in this way. We selected the Contingent model in the main document because it provides cleaner transaction completion, and think it might suggest a variation in the UMM Business Transaction Patterns.*

Also note that the *same* CoordinateCommitments activity interacts with the "Place Offer" requesting activity in both transactions: the one between Carrier and Distributor, and the one between Supplier and Distributor. Look back at **Figure 4** to see all of the Coordinate Commitments object flows.

Decision Logic

Coordinate Commitments evaluates the offers from both Carrier and Distributor against the original order specifications from the end Customer:

- Do we have offers from both Supplier and Carrier?
- Do the product specs in the Supplier's offer satisfy the product specs from the Customer?
- Does the delivery time offered by the Carrier meet the timing constraints from the Customer?
- Is the combined price from Supplier + Carrier acceptable to the Distributor, in relation to the price the Customer will commit to pay?

If the answer to any of those questions is "no", then CoordinateCommitments either needs to try other Suppliers and/or Carriers, or, having exhausted all possibilities, must decline the Customer's order and also send Negative Acknowledgments to whichever of Supplier and Carrier accepted the Distributor's offer.

Repeat warning: that last part, using negative receiptAcknowledgments to revoke a conditional offer, is an abuse of UMM's intentions for that signal. See the non-transactional models or the Appendix for alternatives, although the alternatives lack such a clean transaction completion.

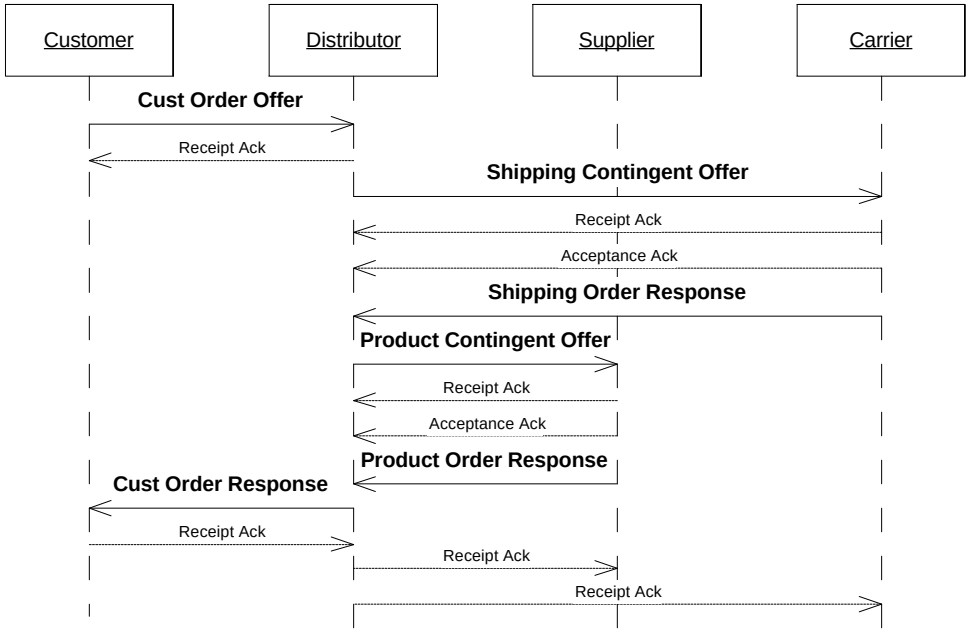


Figure 8: UMM Message Sequence for Drop Dead scenario

Business Action Messages (e.g. Offer and Response) are shown in bold.
Business Signal messages (ReceiptAcknowledgments etc.) are shown in a smaller font with dashed arrows..

Messages from different UMM Business Transactions are *interleaved*:

- First the Customer Order Offer;
- then the Offers and Responses between Distributor and Carrier and Supplier;
- then (based on the responses from Carrier and Supplier) the Distributor responds to the Customer Order.
- Finally, if the Receipt Acknowledgment from the Customer is positive, then the Distributer sends positive Receipt Acks to Supplier and Carrier, and the overall collaboration as well as all of its component transactions succeeds.
- But if the Receipt Acknowledgment from the Customer is negative, then the Distributer sends negative Receipt Acks to Supplier and Carrier, and the overall collaboration as well as all of its component transactions fails.

No separate compensating or cancelling transactions are required. Every offer-acceptance transaction succeeds, or every offer-acceptance transaction fails.

[Note to UMM experts: the above diagram would never appear in a UMM model in this exact form. We're combining information from several UMM diagrams to show the interleaved messages. In reality, the interactions between Customer and Distributor, Distributor and Supplier, and Distributor and Carrier, would be three separate transactions in UMM, related only by the Distributor's CoordinateCommitments activity.]

Comparison of transactional and non-transactional collaborations

The **non-transactional UMM collaboration models** provided several opportunities for something to go wrong:

1. The Offer transaction between Customer and Distributor.
2. The Cancellation or Confirmation transactions between Distributor and Supplier or Carrier.
3. The Acceptance or Rejection transaction between Customer and Distributor.
4. The Confirmation transactions between Distributor and Supplier and Carrier must be completed successfully before the Acceptance between Distributor and Customer. But if the Acceptance transaction fails, another set of Cancellation transactions will be required between Distributor and Supplier and Carrier.

In contrast, the **transactional model** contains:

- One and only one transaction between Customer and Distributor, with no separate Acceptance transaction.
- One and only one transaction between Distributor and Supplier, and Distributor and Carrier.
- The Distributor's response to the Customer Order is dependent on the responses from Supplier and Carrier, and the outcome of the whole set of transactions is dependent on the end state of the transaction between Customer and Distributor.
- No separate compensating or canceling transactions in any case.

BTP Multi-Party Transaction Model for Drop-Dead scenario

BTP is agnostic with regard to the application actions it coordinates. Therefore it can be used with any of the previous versions of the Drop-Dead scenario – indeed any other such scenarios as well.

BTP requires that each application message that is part of a transaction be identified in some manner as being part of that transaction. It does not require that this identification is achieved in any particular manner – this is left to the application protocol designer and explicit or implicit means may be used. However, the BTP specification does provide one standard way of achieving this by adding a CONTEXT message to each application message.

From the business application's point of view, all it needs to do is request the initiating of a transaction, perform its application work then request the confirmation of the transaction. It will then be informed as to whether the transaction completed successfully (confirmed) or failed completely (cancelled).

We can also combine UMM and BTP. Figure 8 below shows the same business scenario as Figure 7 with optimised BTP messages added.

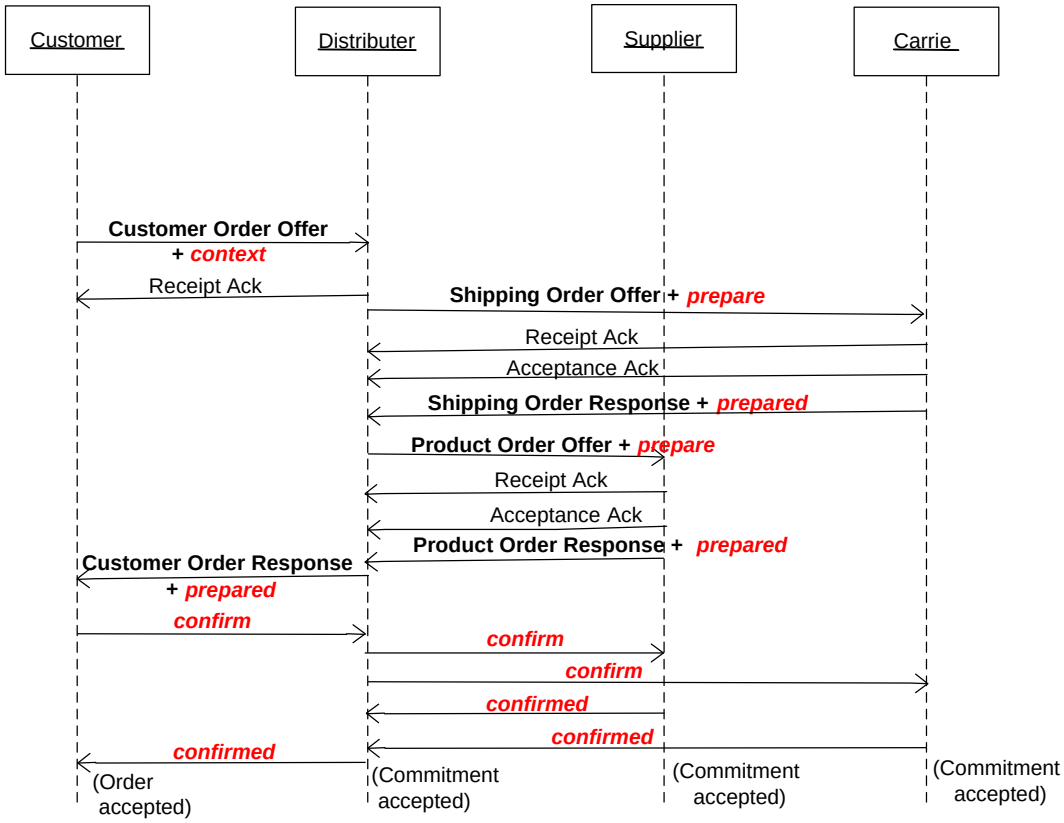


Figure 9: UMM + BTP Message Sequence

The differences between this diagram and Figure 7 include "context", "prepare" and "prepared" messages added to some of the UMM business action messages, and a set of additional "confirm" and "confirmed" messages at the end.

BTP specializes in transaction completion and recovery, and these additional signals are designed to provide extremely reliable completion, or built-in mechanisms for recovery if something goes wrong. We will get deeper into these differences later under the heading **Transaction Completion vs Business Coordination**.

BTP contains many more features than we describe here (and so does UMM). Our focus on this document was on the similarities between BTP and UMM, as applied to one multi-party business scenario, so we have not pursued many of the differences. For full details of either specification, follow the References.

Comparison of BTP and UMM models

As you can see by comparing the Message Sequence diagrams, the message sequences are very similar. And the transactional effects are identical: either all orders succeed, or all orders fail.

UMM also requires that messages be placed in context, although the context would be called BusinessCollaboration and BusinessTransaction.

An additional set of objects appears in the more elaborate BTP diagram shown below, labeled "Customer BTP", "Distributor BTP", etc. These are BTP software components that implement transaction completion and recovery mechanisms. They make this diagram look more complex than the corresponding UMM diagram, but in reality, UMM has another lower layer of interactions, too. The difference is that in UMM, transaction completion is managed by the RequestingBusinessActivity of each transaction, not a separate layer of components. *The overall complexity is similar, just factored differently*.

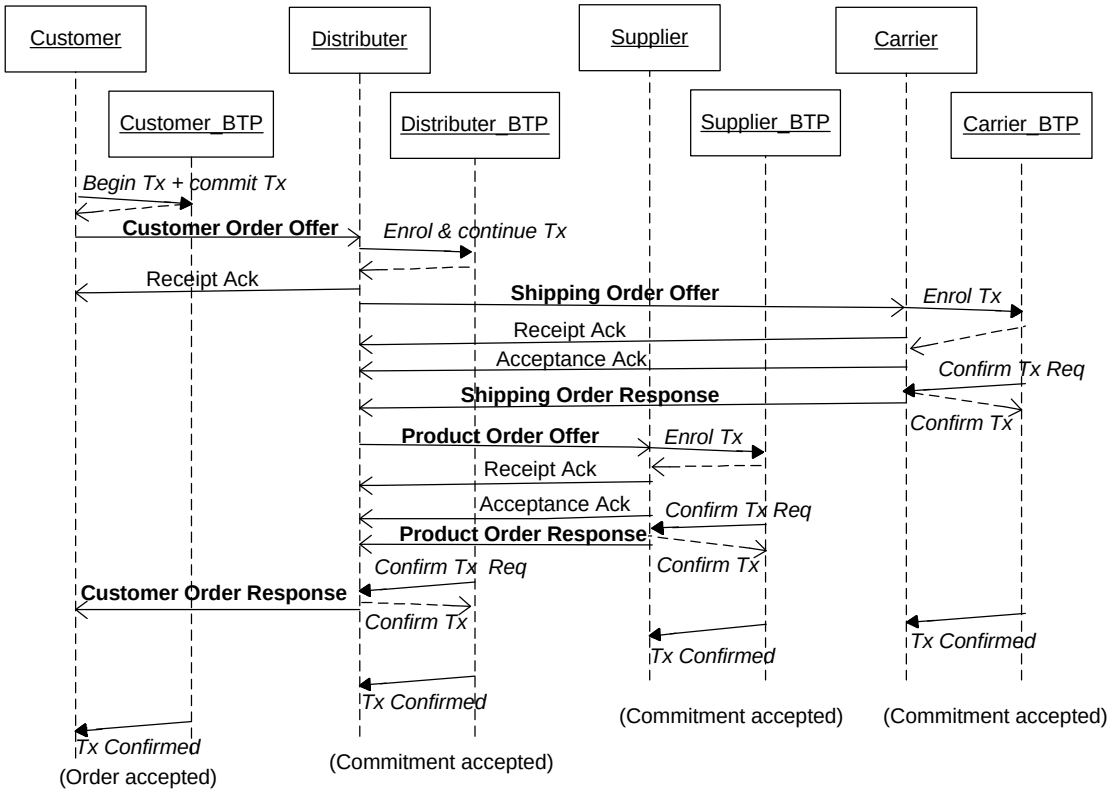


Figure 10: Sequence with BTP Transaction Management Components

Conclusions

Two conclusions should be evident from this document:
1. UN/CEFACT UMM and Business Collaboration Protocol can support multi-party transactional behavior.
2. UNCEFACT BCP, ebXML BPSS and OASIS-BTP can be reconciled.

UMM-BCP can support multi-party transactions

Despite a paragraph in the UMM documentation that seems to say otherwise, we have demonstrated how to use UMM to model multi-party "transactional collaborations".

We have also described other non-transactional approaches to multi-party coordination in UMM. These approaches are similar to Cohesions in BTP.

We do not claim to have solved all problems, but do claim that the approaches here are workable first steps toward implementing multi-party collaborations in UMM that provide transactional or near-transactional behavior.

BCP, BPSS and BTP can be reconciled

If (as promised) UN/CEFACT reconciles BPSS and BCP, then all we need consider here is reconciling BCP and BTP.

BCP and BTP are more alike than many people think. "Many people" once included the authors of this document. But as we explored the problem more deeply, similarities emerged:
- Both BCP and BTP use transactions for external business state alignment.
- Both allow relaxing some of the same ACID transaction rules.
- Both recognize two distinct levels of messaging: protocol signals and business application messages.
- The BCP protocol signals can be mapped to the BTP protocol signals.
- Both prefer two-party transaction contracts (although this was a major stumbling block in earlier discussions, because it seemed like BTP did not).
- Both consider business transaction protocols to be distributed state machines.
- The BTP Coordinator concept is useful in BCP, as we saw from the multi-party Collaboration models.
- The UN/CEFACT Commitment concept is useful in BTP. Commitments determine who does the Coordinating, and what business elements need to be coordinated.
- As we have seen, a very similar overall message sequence can be achieved in multi-party scenarios for both UMM and UMM + BTP.

To sum up, BCP and BTP are members of the same design family. They complement, rather than compete with each other.

Immediate practical consequence:

BTP-compliant runtime software can execute complex UMM Business Collaborations.

Many people have been perplexed about how to get complex UMM Business Collaborations up and running in the real world. Single Business Transactions are easy; they're very close to RosettaNet PIPs. But the more complex collaborations are not fully specified yet even in UMM: that is, the UMM Business Collaboration Protocol is still a work-in-progress. BTP runtime software is coming on the market now, and is a capable option for now and the future. (See next topic.)

Suggestion for next stage of UMM BCP project:

Harmonize BCP with BTP State Machines.

Actually, this suggestion is already in motion: Tony Fletcher of BTP is a member of the UMM BCP work group, and this document represents some fruits of the cross-group collaboration. We just want to underscore the opportunity here.

The UMM Business Collaboration Protocol project is in process of defining state machines for executing business transactions and collaborations. BTP has already defined state machines that achieve many UMM requirements.

BTP has focused only on the transaction completion interactions, whereas UMM has focused more heavily on business coordination than transaction completion.

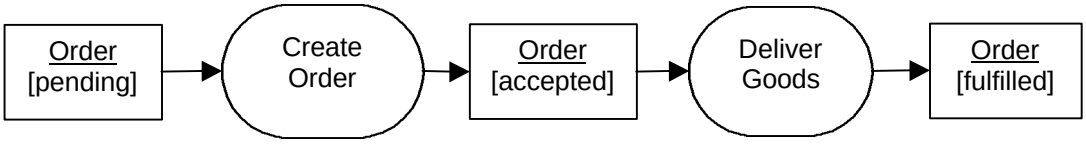
Putting these two specialties together could not only accelerate the completion of the UMM BCP, but also accomplish a high degree of convergence of BCP and BTP.

Convergence of BCP and BTP would be one step toward converging all of the competing business transaction protocols.

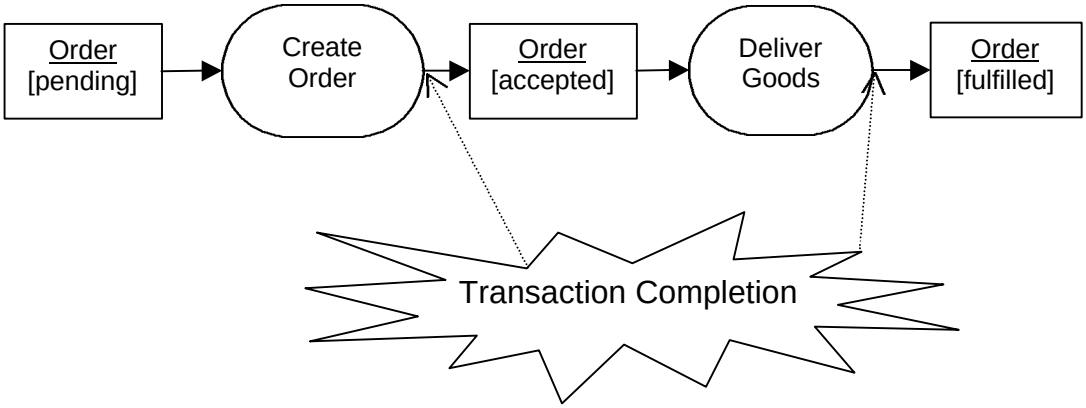
Transaction Completion vs Business Coordination

UMM Business Collaborations use the states of business objects like Orders, Commitments, Deliveries and Payments as preconditions for activities, and postconditions of business activities are represented as subsequent states of business objects. That's the concerns that we're calling Business Coordination in this section of the discussion. Our intent is to contrast

- Business Coordination using the states of business objects, from
- Transaction Completion, or reliably changing the states of business objects.



Business objects like Orders represent the state of a business deal. Transactions change the states of the business objects in a business deal. In a distributed business transaction, both Buyer and Seller will have a version of the same object. You might call each of their versions half-an-object, because they are both supposed to be versions of one and the same order. The transaction protocol's job is to keep both halves of the order in sync. That's called "business state alignment". We don't want the Seller think the order was fulfilled, while the Buyer thinks something is wrong and it is unfulfilled.



That is why trustworthy transaction completion is so important.

Business coordination is also important. The order being in the [fulfilled] state is the precondition for invoicing and payment (in this little scenario). Preconditions ensure that no transaction is executed before its time.

So transaction completion and business coordination are both important, and must dance together. (More like a tango than a mosh pit, we hope.) (No wonder business coordination is often called "choreography".)

As we stated above, BTP has focused exclusively on transaction completion, while UMM-BCP has focused on both business coordination and transaction completion. As we saw in Figure 8, BTP adds some signals to the UMM message flow.

For a simpler picture, here is a comparison of message flows in a two-party transaction:

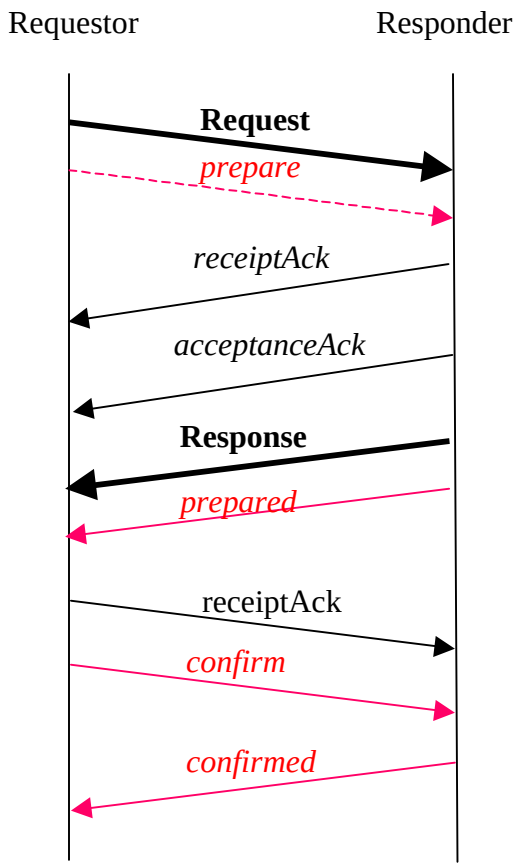


Figure 13: UMM vs BTP message flows

UMM = black, BTP = red.

However, we can piggyback some of the BTP messages onto UMM messages to save roundtrips, like so:

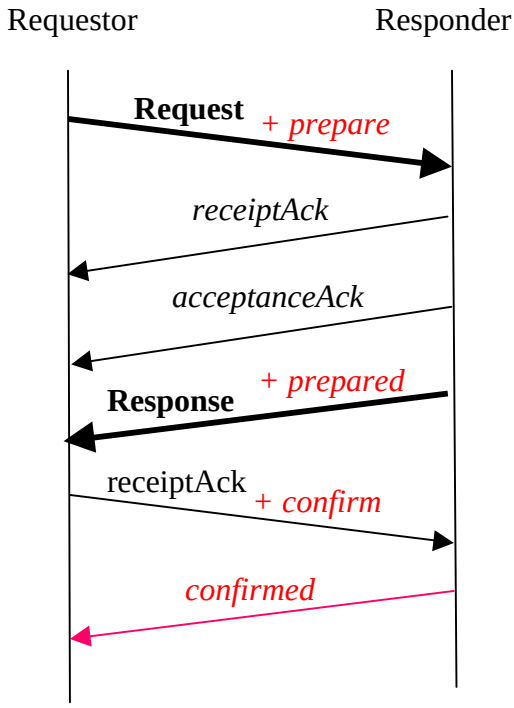


Figure 14: Piggybacked messages

Those additional signals accomplish BTP-style transaction completion. BTP separates transaction completion into specialized software components. "Prepare" tells the BTP transaction components to get ready to complete a transaction. "Prepared" is a response from participants that they're ready. The "confirm" signals tell all participants that the transaction is completing successfully. The "confirmed" signals report to the coordinator that the confirmations have been received and acted on, and thus that everybody knows the transaction succeeded.

So here's a comparison of the concerns of UMM-BCP and BTP: where they have exclusives, and where they overlap:

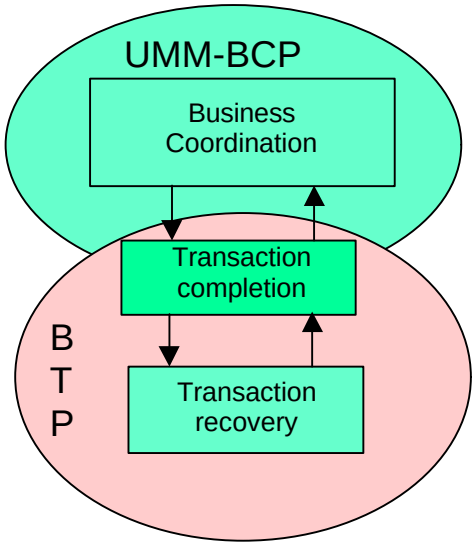


Figure 15: Comparison of concerns

UMM-BCP is solely responsible for Business Coordination; BTP is happy to leave that to somebody else. Both UMM-BCP and BTP have features for Transaction Completion, although that is BTP's specialty, and it has more. BTP alone has built-in features for Transaction Recovery, in keeping with its specialty in Transaction Completion.

We recommend that in the upcoming UMM-BCP work, the additional steps in Transaction Completion and Recovery in BTP be considered objectively, with test cases and risk assessments. Moreover, we recommend that the BTP group consider the UMM-BCP Business Coordination approach as a "business application" layer that will fit right into the BTP specifications.

As we've seen, both UMM-BCP and BTP were designed by smart people who have deep experience in very similar problem domains. We think it will be good for both groups to try to compare and combine all of this intelligence.

We hope this document will help.

Human and Conceptual Relationships of Standards Groups

UN/CEFACT BCP, ebXML BPSS and RosettaNet

These three specifications and organizations are related. Moreover, they were all originally designed for distributed or networked business transactions, not database transactions.

RosettaNet came first of the three, and was in turn based on prior work in the TeleManagement Forum called the Telecom Operations Map (TOM), and UN/ECE recommendations on international ecommerce (see References).

RosettaNet transaction contracts are codified as Partner Interface Processes, or PIPs. PIPs are two-party business transaction protocols.

UN/CEFACT adopted a Business Collaboration Framework metamodel derived from RosettaNet. The UN/CEFACT Business Collaboration Protocol (BCP) uses a two-party Business Transaction model that follows RosettaNet PIPs very closely.

BCP then adds higher layers called Business Collaborations, which may be composed of, and coordinate, many Business Transactions. BCP does *not* promise transactional behavior in Business Collaborations. In other words, BCP promises 2-party and only 2-party transactions. ***(However, this document will demonstrate that it is* possible *to implement transactional behavior in multi-party BCP Collaborations.)***

UN/CEFACT BCP is intended to be technology-independent. It could be implemented in many different ways, using different techologies, ranging from manual to highly automated.

ebXML introduced the Business Process Specification Schema, or BPSS, which is represented as an XML document following an XML Schema or DTD. BPSS is intended to be an XML implementation of the UN/CEFACT BCP.

All of the above is somewhat oversimplified in that there are some differences between BPSS and BCP and RosettaNet PIPs. But the relationships are strong – several of the same people worked on all of the specifications - and the various organizations are working on resolving the differences.

UN/CEFACT BCP and the Business Collaboration Framework metamodel are incorporated in the UN/CEFACT Modeling Methodology, or UMM.

OASIS BTP

The Business Transaction Protocol specification also has a long pedigree in the history of distributed transaction protocols.

The IBM SNA (System Network Architecture) was among the first commercial products to include a distributed transaction protocol. This was part of the facilities of LU6.2. However, although LU6.2 was widely used, the full transaction capability, using the two-phase commit of synchronization points, was not often configured.

In the Open System Interconnection standards effort (a co-operative standards effort between ISO and the ITU), the Commitment, Concurrency and Recovery (CCR) standards specified a two-phase commit protocol, which was used in the OSI Transaction Processing standard, which was itself functionally very close to SNA LU6.2. Tony Fletcher and Peter Furniss of the BTP Committee worked on both CCR and OSI TP – Peter was editor of both protocol documents. (Tony, of course, is one of the authors of this document,

and Peter was an important contributor.) OSI was also a major influence on the Telemanagement Forum, RosettaNet, and UN/CEFACT BCP.

The Object Transaction Service (OTS) specification provides two-phase commit support in the CORBA world – Alastair Green and Peter Furniss were involved in implementation and some of the later revision of OTS. Peter was also involved with TIP, the "Transaction Internet Protocol", another open two-phase commit protocol specification, which was published as an Internet RFC.

The advent of XML gave an opportunity to re-examine transaction protocols and the feature mix. One early attempt was XAML. For better or worse this foundered and some of the parties transferred to start a new group in OASIS, which became the Business Transaction Protocol Technical Committee. The work started in earnest in March 2001 with major contributions from HP Arjuna, BEA and Choreology. There was identifiable, if undocumented, influence on the development of BTP from the standards mentioned above – especially OTS, TIP and CCR – but with several innovations, mostly motivated by the desire to make BTP applicable to more open or inter-organanization environments. BTP version 1.0 was formally agreed at a meeting in May 2002 for public release 3 June 2002 (to allow time for a primer to appear at the same time). Choreology played a very major role in the development of BTP – Peter Furniss was editor, and most of the specification was authored by him, Alastair Green and Bill Pope (who was chairman of the BTP committee).

The (potential) commercial importance of this area is demonstrated by the fact that the story does not end here. In August 2002 Microsoft, IBM and BEA issued their own Web Service oriented TP specifications – WS- Coordination and WS-Transactions (often known as WS-C/T), along with a business process specification language BPEL4WS.

What we learn from this history

All of the major current electronic business transaction protocols are related by common ideas and participants.

They are all descendants of LU6.2 and OSI TP. Several people carried ideas from one project to the next. Thus the specifications share many concepts, including:

- distributed network transactions
- managed objects
- state alignment using distributed state machines and protocol stacks
- contract interfaces
- global transaction identifiers

In other words, a strong basis exists for converging all of these efforts into one.

Convergence between UN/CEFACT BCP and OASIS-BTP

UN/CEFACT and OASIS have a Memorandum of Understanding, and the recently completed ebXML project was a collaborative effort of both organizations.

Several conversations took place between ebXML Business Process Work Group and BTP members over the last year, to clarify the differences between their approaches.

Tony Fletcher of BTP joined ebXML in March 2000, and is now a member of the UN/CEFACT Business Collaboration Protocol and Architecture work groups. He has been working out the details of how BTP transactions could fit within the UN/CEFACT architecture.

Very recently, Bob Haugen of the UN/CEFACT Business Process work group met with several BTP participants, including Tony Fletcher and Peter Furniss. This document is a direct result of that meeting.

Evolution of this document

The ideas began to form in a discussion thread on the UNCEFACT Business Collaboration Protocol mailing list. The focus of the thread was to attempt to reconcile differences in transaction concepts between OASIS-BTP and UNCEFACT-BCP.

Choreology people were conducting their own discussion at the same time, about how to support UMM models with BTP software.

Then Haugen visited Choreology and continued the discussion with Fletcher, Alastair Green, Peter Furniss, Mike Leznar and Bill Pope, among others. The ideas for modeling multi-party transactions in UMM crystallized during the meeting at Choreology.

Haugen and Fletcher then summarized the discussions in the form you see here, with help from many of the others named above.

We expect the evolution to continue with attempts to reconcile other electronic business transaction protocols, until order emerges from the current chaos. “Order” could mean an accepted standard, or a small number of standard protocols for clearly different problem domains.

Related Work

Jean-Jacques Dubray has written (and continues to write and collect) comparisons of the major electronic business process specifications on his Web site http://www.ebpml.org.

References

BTP
OASIS Business Transactions Committee:
http://www.oasis-open.org/committees/business-transactions/
Choreology Ltd., BTP software:
http://www.choreology.com/

Commercial Law
UNIFORM RULES OF CONDUCT FOR INTERCHANGE OF TRADE DATA BY TELETRANSMISSION (UNCID)
http://www.unece.org/trade/untdid/texts/d220_d.htm
UN/ECE Recommendation No.26, THE COMMERCIAL USE OF INTERCHANGE AGREEMENTS FOR ELECTRONIC DATA INTERCHANGE,
http://www.unece.org/trade/untdid/texts/d240_d.htm
UN/ECE Recommendation No.31, ELECTRONIC COMMERCE AGREEMENT
http://www.unece.org/cefact/rec/rec31/rec31_2000_00tr257.pdf

ebXML http://www.ebxml.org/
BPSS http://www.ebxml.org/specs/ebBPSS.pdf
CPPA
http://www.oasis-open.org/committees/ebxml-cppa/documents/ebcpp-2.0.pdf
http://www.oasis-open.org/committees/ebxml-cppa/
Contract Negotiation
http://www.ebxml.org/specs/bpPATT.pdf

REA (Commitments)
William McCarthy's REA papers: http://www.msu.edu/user/mccarth4/
With speech acts, courtesy of the OpenebXML project:
http://www.openebxml.org/information/papers/ecomo02stockholm.pdf

RosettaNet http://www.rosettanet.org

UN/CEFACT http://www.unece.org/cefact/
UN/CEFACT's working groups are being reorganized. The URLs below are temporary.
UMM http://www.gefeg.com/tmwg/n090r10.htm
Business Collaboration Protocol
http://homepage.mac.com/knaujok/TMG_Files

WS-T
http://www-106.ibm.com/developerworks/webservices/library/ws-transpec/