New Capabilities for Interaction Modeling in BPMN 2.0

Conrad Bock, National Institute of Standards and Technology, USA, and Stephen A. White PhD, International Business Machines

1. Introduction

Interaction models capture how businesses interact with customers and each other to provide products and services. Models are needed to reach agreements about what will be provided to whom and when, and to gather requirements and expertise in one place for a successful business. The trend towards combinations of products and services increases the complexity of interactions beyond the capacity of conventional business process modeling languages. Conventional process modeling typically focuses on business internals. Interaction models hide the proprietary aspects of business processes, while exposing those aspects needed for interaction. They scale to complex interactions between many parties, as in supply chains.

This paper provides a high-level introduction to new features of interaction diagrams and interactive processes in the Business Process Model and Notation (BPMN) Version 2.0 [1]. It covers the Choreography diagram introduced in BPMN 2, improvements to the Collaboration diagram from BPMN 1, including Conversations, as well as enhancements in process modeling to support interactions. The paper assumes familiarity with earlier versions of BPMN. Section 2 explains why interaction models are needed for modern business functioning. Section 3 covers interaction models in BPMN 2. Section 4 describes added capabilities for interactive processes needed to deploy interactions. Section 5 summarizes BPMN 2 support for interactions and interactive processes. A companion paper introduces new features for process diagrams in BPMN 2 [2].

2. Business Motivation for Interaction Models

Modern businesses typically provide services along with products ("solutions"), and partner with other businesses in delivering solutions.[1] For example, producers of cell phones provide mobile services, many products are paired with maintenance contracts, and shipping is usually provided by separate businesses. Services can increase potential markets by providing multiple services on a single product. New services can help differentiate businesses from competitors. They also enable businesses to focus on core value, while partnering for non-core services.

Business services are characterized by interactions between businesses and their customers and partners. Businesses have a wide range of involvement with customers in providing services, from help desks to on-site project sup-

[1] The term "service" is used in the business sense, rather than a web service or other software operation. Business services can be complicated to deliver, and require more information to specify than simpler software services such as getting stock quotes. Antoine Lonjon provided some of the points in this paragraph.

port. Sometimes multiple businesses interact with customers when providing the same service, for example when a credit agency is involved with a purchase. The interactions might be very short from start to end, or take place over a long period. They might have just a few exchanges between customers and businesses, or very many. The items exchanged might be only information, include goods, or involve movement of personnel during the interaction.

Service interactions require agreement by the businesses, customers, and partners involved. For example, they determine what information or goods are needed by whom and at what time, when to expect personnel to be involved and what they will do, how complaints and unusual situations are handled, and whether followups are scheduled in advance. These agreements might be completed in advance, or have details worked out during the interaction, as in case management. They might define what happens if the agreement is not followed for some reason, including compensating interactions.

Capturing service interactions in diagrams helps clarify agreements between the participants, gather requirements and solution expertise in one place, and facilitate coordination between the parties when the interaction is carried out. This helps lower costs for providers and consumers, reduce unnecessary or unsatisfactory interactions, and identify new areas for service development, including combining and adapting existing services. Multiple diagrams can be developed for the same service to evaluate alternative interactions, or for negotiation with different potential business partners.

Interaction diagrams for reaching agreements hide proprietary aspects of business processes carrying out interactions. This is different from typical business process models, which usually show all the details necessary to carry out a process. Languages for interactions enable modelers to separate out portions of business processes needed specifically for interactions, while keeping the rest private. These can be linked together to specify how private processes support public ones exposed for interactions.

3. INTERACTION DIAGRAMS

BPMN 2 has two diagrams for interactions: Collaboration and Choreography. The first is available in BPMN 1.x, and enhanced in BPMN 2, while the second is in BPMN 2 only. The diagrams show different aspects of interactions, sometimes using different notations for the same concepts, or highlighting some concepts over others. Section 3.1 covers the concepts in common to both diagrams, while Sections 3.2 and 3.3 describe concepts available in only one of the diagrams.

3.1 Interaction Basics

Interaction diagrams in BPMN 2 have these elements in common:

- Participants are the interacting agents. These might be businesses, departments, or people, for example, or automated agents in software or hardware.
- Messages are sent between Participants. These can be informational or physical, including physical things that do not carry information, such as cars or furniture.
- Messages Flows occur at certain points during the interaction, between particular Participants. The same Message can be carried by more than one Message Flow.

Figure 1 shows the notations for these three basic concepts. Participants in Collaboration diagrams on the left are shown with rectangles (called "pools"), while the Choreography diagram on the right shows them as bands inside a rounded rectangle, called a Choreography Activity. Collaboration diagrams show Participants much more prominently than Choreography, so are useful when relationships between Participants are the primary concern.

Messages are shown as envelopes on both interaction diagrams, with a label naming them. Message Flows show which Participants exchange the Messages. In Collaboration diagrams Message Flows appear as dashed arrows with Messages optionally overlaid on them. In Choreography, Message Flows are shown as Choreography Activities, with Messages linked to them by dotted lines called Associations. The unshaded bands of Choreography Activities are Participants sending the Message, and shaded bands are the ones receiving them.

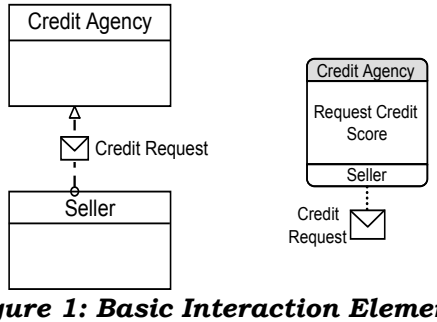

Figure 1: Basic Interaction Elements

Participants can be "roles" in the interaction, such as Seller or Buyer, or individual entities, such as Walmart or the U.S. Government. When Participants are defined as roles, the individual entities participating ("playing" the roles) can be different each time the interaction is carried out (this can be determined by other interactions, see Section 3.3). Interactions with role-based Participants are more widely applicable, but in some cases, interactions between specific individual businesses or organizations are needed.

Participants can represent things of any size, from entire industries to individual people. Modelers choose the appropriate level of granularity, which can vary between diagrams. Large Participants can be managed by indirectly nesting other Participants in them, through nested interactions and Processes, see Section 3.3.

Participants in Collaborations can be used in a "black box" way hiding all their internals, in a "white box" way exposing all their internals, or "grey box"

showing some internals. For completely separate businesses, black or grey box Participants are usually more appropriate, whereas Participants within a single business might be white box, depending on how independent the various departments or subsidiaries are. In BPMN 2, Participant internals are Processes responsible for carrying out interactions, see Section 4.1. Modelers can choose how much internal Processes to expose based on their application, see public Processes in Section 4.2. Regardless of how visible internal Processes are in interaction diagrams, Processes as they are carried out can only be affected by other Participants through Messages received, not by Process Activities occurring in the other Participants.

Messages usually describe the kinds of things flowing between Participants, rather than all the details of actual things flowing at particular times. For example, a Message might be an agreement document between two companies, and will typically appear in the model without all the details of a particular agreement. The actual agreements sent between companies will vary depending on the particular companies and the nature of the agreement. Since Messages describe only some aspects of the things flowing, the same Message can appear on multiple Message Flows or Choreography Activities, and in multiple diagrams, even though the actual things flowing when the interactions are carried out will probably be different each time the Message is sent. In some cases the actual thing flowing might be the same across Message Flows, for example, the same actual document might be forwarded unchanged between Participants.

Businesses agreeing to interactions might allow other Messages to flow between them that are not specified in the interaction diagrams. For example, businesses might agree to the most prominent Messages sent between them, and capture these in interaction diagrams, but allow more detailed messaging to occur when the interactions are actually carried out. In BPMN 2, interaction diagrams allowing more Messages to flow between Participants than specified in the diagrams are called *open*, otherwise they are *closed*. The interactions in Figure 1 are not intended to be complete, but if they were, they would probably be open to allow other Messages to flow when the interaction is carried out.[2] Interaction diagrams are assumed to be open, unless the modeler specifies otherwise. Interaction diagrams do not visually show whether they are open or closed, but this information is accessible in other ways with tools supporting interaction diagrams.

3.2 Message Flow Sequence

Messages usually flow between Participants in a particular order. For example, in many retail purchasing interactions, payment is made before the product is delivered. Choreographies capture this most directly, as Sequence Flows between Choreography Activities (see Figure 2). Sequence Flow arrows indicate that the Message in the Choreography Activity at the tail of the arrow flows before the Message in Choreography Activity at the head. In this example, the Message requesting a credit score is sent before the one providing the credit score. The sending Participant in the first Message Flow is the

[2] Closed interaction diagrams do not prevent business entities from interacting based on participation in other interaction diagrams. Businesses might participate in many interactions specified by many diagrams, some open and some closed. They can send and receive unspecified Messages as part of carrying out open interactions, but not as part of carrying out closed ones.

receiver in the second, capturing a simple request-response interaction. The response may be a very long time after the request, or a very short time, depending on how much processing occurs within the receiver of the first Message.

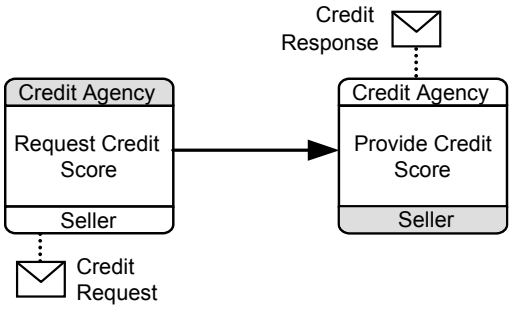

Figure 2: Message Flow Sequence in Choreography

Choreography diagrams can prevent additional Messages from flowing during the time between particular sequential Choreography Activities, even if they are open overall, see Section 3.1. This is done with *immediate* Sequence Flows, which prevent Messages from flowing during the time between Choreography Activities, unless other Activities are explicitly specified on parallel paths, see Parallel Gateway below in this section. If the Sequence Flow in Figure 2 were immediate, no other Messages could flow after the Credit Request Message and before the Credit Response. Immediate Sequence Flows do not require the time between Activities to be zero, they only prevent unspecified Message Flows from occurring when the Choreography is carried out. Choreography diagrams do not visually show which Sequence Flows are immediate, but this information is accessible in other ways with tools supporting Choreography diagrams.

Collaborations can indirectly capture Message Flow sequence by including Processes in Participants, see the example in Figure 3. It shows a Collaboration with the same Participants as the Choreography in Figure 2,[3] but has a Process in the Seller. Message Flows in Collaboration diagrams can link to Process Activities. In Figure 3, the Message requesting a credit score flows from a Process Activity sending that Message, indicated with a filled envelope adornment on the Activity. The Sequence Flow leads to a receiving Activity, indicated with an empty envelope adornment, which has a Message flowing into it providing a credit score. This gives the same order of Message Flows as Figure 2.

[3] Participants in different interaction diagrams are separate, even if they have the same name, enabling Choreographies and Collaborations to be defined separately and linked together later. If Participants in different diagrams are intended to be the same, they must be linked together by Participant Associations, see Figure 7 and Figure 9. Tools can create Participant Associations automatically based on Participant names or other criteria.

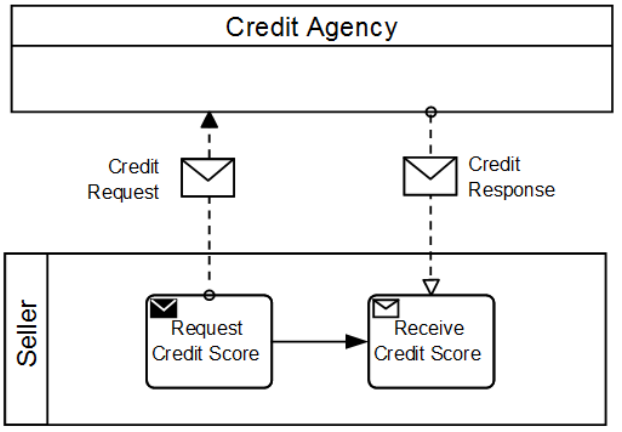

Figure 3: Message Flow Sequence in Collaboration

Choreography diagrams can be overlaid on Collaborations to show their relationship, with each Choreography Activity overlaid on a corresponding Message Flow in the Collaboration. This is useful if the Choreography and Collaboration are developed separately and need to be checked for consistency. Choreography diagrams can be overlaid on Collaborations that do not show any internal Processes, as a way of highlighting Participants more than Choreographies would by themselves. Message Flows and Participants in the Choreography and Collaboration elements are linked together with Message Flow Associations and Participant Associations, respectively. Message Flow Associations do not have notation, but they are indicated by the graphical overlap of Choreography Activities and Message Flows in the Collaboration. Participant Associations do not have notation either, but are accessible in other ways with tools supporting interactions. Showing Choreographies inside Collaborations can be challenging with more than two Participants.

Message Flow sequencing is restricted by the limited visibility Participants have into each other (see Sections 2 and 3.1 regarding Participant visibility). In particular, Participants cannot send Messages based on Process Activities occurring in other Participants. For example, the top of Figure 4 shows an invalid Choreography, because the Message Flow of the last Choreography Activity assumes the Seller can tell when the Credit Agency sends a Message to the Bank. The Process in the Seller would need to wait for the Process in the Credit Agency to send a Message to the Bank, but the Seller cannot know when that happens.[4] The problem is addressed at the bottom of Figure 4 using a Parallel Gateway. This enables Activities after the Gateway to happen without sequencing between them (more about Gateways below). The top Activity after the Gateway can proceed because the Credit Agency knows when it receives a Message from the Seller, while the bottom Activity after the Gateway can proceed because the Seller knows when it is finished sending the Message to the Credit Agency. The Seller does not need to wait for the Request Credit Score Message to arrive at Credit Agency, which cannot be

[4] An emerging technology enables Participants to see when other Participants send and receive Messages [3]. The top Choreography in Figure 4 would be valid using these platforms, because the Seller can tell when the Credit Agency sends a Message to the Bank. However, coordination platforms are not mature enough for BPMN 2 to assume such a high level of Participant visibility.

known to the Seller anyway.[5] The general rule for Message Flow sequencing, and Choreography Activity sequencing in particular, is the sender in Choreography Activity must also be a Participant in the immediately previous Activity, as either sender or receiver (the sender in the first Activity is the initiator of the whole Choreography). The second Activity at the top of Figure 4 follows this rule, but the third does not. All the Activities on the bottom of Figure 4 follow the rule.

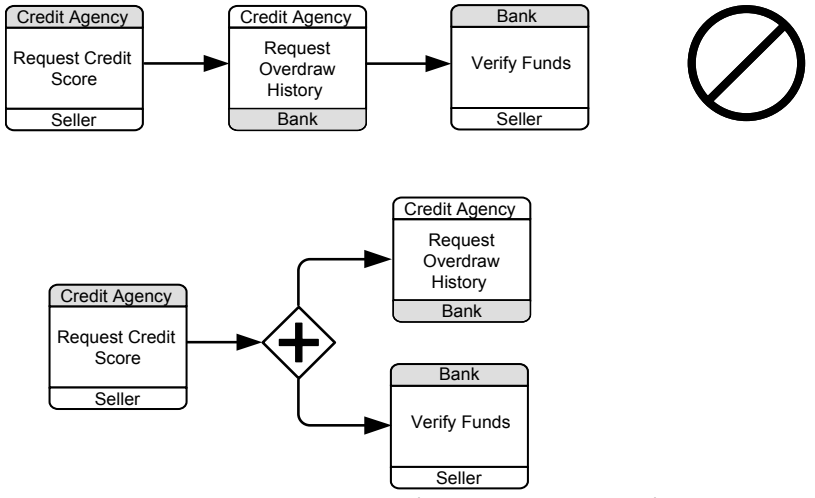

Figure 4: Message Flow Sequence Rules

Choreography diagrams can use Gateways to split and merge Sequence Flows in similar ways as BPMN Process diagrams, but with some restrictions due to limited Participant visibility. The rule above about Participants in sequences of Choreography Activities applies even when Gateways are present. The sender in a Choreography Activity must also be a Participant in the immediately previous Activity, either as sender or receiver, even if Gateways are interposed between the Activities. For split Gateways, which have a single Sequence Flow coming in and multiple going out, all senders in Activities after the Gateway must participate in the Activity before the Gateway. For merge Gateways, which have multiple Sequence Flows going in and a single one going out, the sender in the Activity after the Gateway must participate in all the Activities before the Gateway. The Participant rules also apply to chains of Gateways with no Choreography Activities in between. The Activities immediately before and after the chain follow the Participant rules. Additional Participant rules apply to most of the specialized Gateways, see below. The notation for Gateways in Choreography diagrams is the same as in Process diagrams, see Figure 5.

[5] Normally Sequence Flow only applies when an Activity is finished, and in Choreography this would mean the Message Flow from Seller to Credit Agency is complete and the Message been received before the Message from Seller to Bank is sent. This would invalidate the Choreography at the bottom of Figure 4, because the Seller cannot know when the Credit Agency receives the Message. However, for purposes of checking Message Flow sequences, it can be assumed that Messages take no time to be transmitted between Participants [4]. In the Choreography at the bottom of Figure 4, the Message from the Seller to the Credit Agency can be assumed to take no time, so the Seller can send the Message to the Bank right after it is finished sending the Message to the Seller.

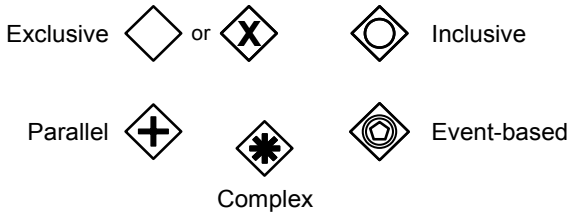

Figure 5: Gateways

Exclusive Gateways that split flows in Choreography are useful when a decision will be made about what Message is sent next, and between which Participants. Participants in Choreography Activities immediately following the Gateway must have sent or received Messages with the data used in the decision sometime earlier, and the data must not have changed before the decision is made. This enables all senders to make the same decision about whether to send a Message, and receivers to avoid waiting for a Message that never arrives. Choreographies cannot store this data, because interactions have no controller apart from the Participants. Exclusive Gateways that merge flows do not synchronize the incoming paths, which means the Message represented by the Activity following the Gateway is sent as many times as there are incoming flows. No additional Participant rules are needed for merging Exclusive Gateways, because no additional decisions or other coordination occurs between Participants immediately before and after the Gateway.

Parallel Gateways in Choreography diagrams split and merge sequences in separate paths, without introducing any additional sequencing between Activities in the separate paths. All outgoing paths are taken in parallel splits when the Choreography is carried out, and incoming paths are combined in a merge when they are all completed. No additional Participant rules are needed, because no additional decisions or other coordination occurs between Participants immediately before and after the Gateway. The sender in the Choreography Activity immediately after the Gateway will be able to synchronize the merge, because it participates in the Activities immediately before Gateway.

Inclusive Gateways are “in between” parallel and exclusive, enabling some or all parallel paths to be split or merged. Participant rules are similar to Exclusive Gateways, because a decision must be made about which paths will be taken, even if it turns out all are taken. Participants in Choreography Activities immediately following the Gateway must have sent or received Messages with the data used in the decision, and the data must not have changed before the decision is made. This enables all senders to make the same decision about whether to send a Message, and receivers to avoid waiting for a Message that never arrives. Choreographies cannot store this data, because interactions have no controller apart from the Participants. Complex Gateways are similar to inclusive but enable modelers to specify when merges happen, for example, to merge when three out of five of the Activities immediately preceding the Gateway are complete.

Event-based Gateways in Choreography split Sequence Flows similarly to Exclusive Gateways, except the data used to make the decision is visible only to one Participant. Choreography Activities immediately after the Gateway must all have the same sender or all have the same receiver, or both. If the senders are all the same, the decision is made by the sender, and the receiv-

ers have internal Event-based Gateways, possibly in parallel flows or with timeouts to accommodate Messages that never arrive. If the senders are different in Activities immediately after the Gateway, the receivers must all be the same, and the receiver has an internal Event-based Gateway to handle whichever Message is sent.

3.3 Grouping Message Flows

Grouping Message Flows helps manage complicated interactions by gathering multiple flows together under a single element.[6] Choreography diagrams support this with Activities representing multiple Message Flows (see examples on the left in Figure 6). These have two Message Flows, as indicated by the Message icons linked to the Participants sending them on the upper right. In this example the Seller sends a Credit Request Message, and the Credit Agency sends back a Credit Response. The thickness of Choreography Activity borders indicates whether the grouped Message Flows can be nested by multiple Choreography Activities, either in the same Choreography diagram or different ones. A thick border indicates they can be used in multiple Activities, while a thin border indicates they cannot. Thick-bordered Activities are Call Choreography Activities. The one on the lower left in Figure 6 is "calling" a Global Choreography Task that groups Message Flows for nesting by multiple Activities and diagrams. The thin-bordered Activity on the upper left is a Choreography Task, which groups Message Flows without a making them available for multiple Activities or diagrams (they are "local" to the diagram). Message icons can only be linked to Choreography Tasks. Message icons for receiving Participants are shaded to match the bands to which they are linked.

The two Choreography Activities on the left in Figure 6 do not explicitly capture sequencing of the Message Flows they group, because they do not expand to another Choreography diagram, as indicated by the absence of small plus-sign markers. However, they can represent two Message Flows at most, and the first Message to be sent is identified by linking its icon to the unshaded Participant for Choreography Tasks. These two Activities combine request and response flows between the two Participants, with the request happening first because it is the initiating Participant, and the response happening next, because it is the only one left.

[6] Grouping Message Flows is also used to specify correlation information, see Section 4.3.

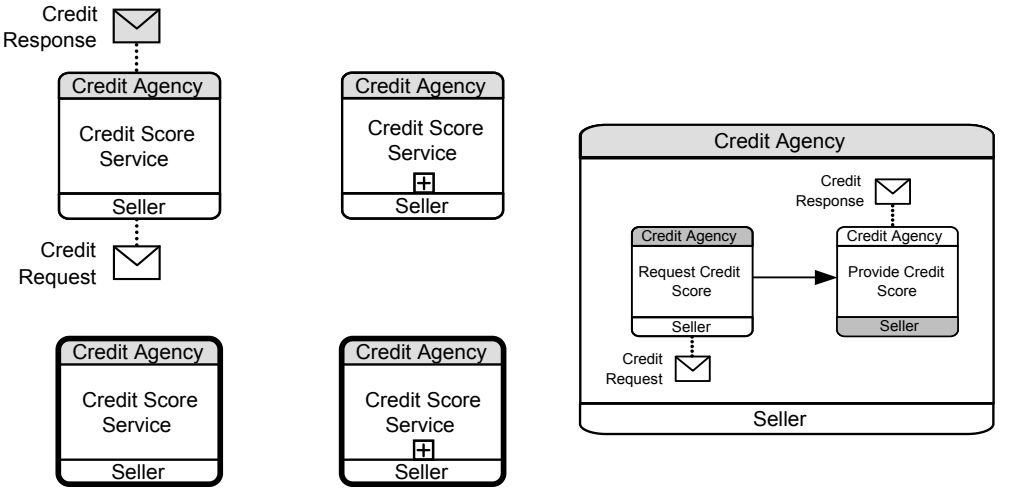

Figure 6: Grouping Message Flows

To capture sequencing of grouped Message Flows, Choreography Activities can expand to entire Choreography diagrams, as illustrated in the middle and right in Figure 6.[7] The middle shows the collapsed forms, as indicated by the small plus-sign markers. In the expanded form shown on the right, they contain a full Choreography diagram, including Sequence Flows between nested Choreography Activities (tools can show nested Choreography diagrams separately from the Activities they are in, to have more space). The Activity in the lower middle is calling the Choreography diagram on the right, which is available to be reused in multiple Activities and diagrams. The Activity in the upper middle is a Sub-Choreography, which nests the diagram on the right without making it available for multiple Activities or diagrams. The expanded form in Figure 6 has a thin outer border, but it can be thick when shown embedded in another Choreography that calls it.

Choreography diagrams sometimes call others that have different Participants, for example Figure 7 shows an expanded call Choreography Activity, adapted from Figure 6. The nested Choreography has the same Participants as Figure 6, but the outer Choreography has Participants Retailer and Financial Services, as shown in the Activity bands. The modeler must specify which Participants in the nested Choreography should be the same as which in the outer one. This is done with Participant Associations, as mentioned in Section 3.2. These Associations do not have a notation in Choreography diagrams, but are accessible in other ways with tools supporting them. In this example, the modeler uses Participant Associations to link Retailer in the outer Collaboration to Seller in the nested one, and Financial Services to Credit Agency.[8]

[7] BPMN 2 supports multiple visualizations of related content, including the collapsed and expanded notations of Choreography Activities. The visual aspects of diagrams, such as shapes and positions, are captured separately from the BPMN concepts they depict, enabling the same concept to be visualized in different ways. These two aspects of BPMN models can be stored and interchanged together.

[8] Participant Associations are needed even if the linked Participants have the same names, because Participants in different interaction diagrams are separate, see footnote 3.

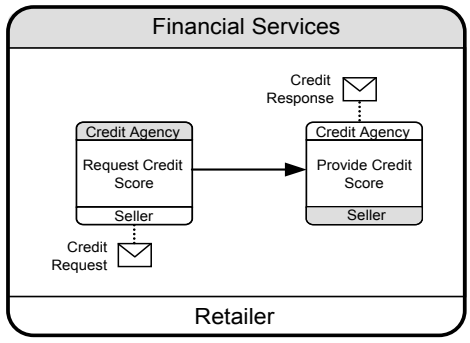

Figure 7: Participant Associations in a Call Choreography Activity

Collaboration diagrams also support grouping Message Flows to manage complicated interactions, using Conversations to stand in for multiple Message Flows (see examples on the left in Figure 8). Conversations appear as hexagons with solid double lines ("pipelines") connecting Participants. The examples expand to two Message Flows shown in the middle left, the Seller sending a Credit Request Message to the Credit Agency, which sends back a Credit Response (related content can have multiple visualizations, including the collapsed and expanded notations of Conversations, see footnote 7). The thickness of Conversation borders indicates whether the grouped Message Flows can be reused in multiple Conversations, either in the same outer Collaboration diagram or different ones. A thick border indicates they can, while a thin border indicates they cannot. Thick-bordered Conversations are Call Conversations. The one on the lower left in Figure 8 is "calling" a Global Conversation that groups Message Flows for reuse by multiple Conversations and diagrams. The thin-bordered Conversation on the upper left groups Message Flows without making them available for multiple Conversations or diagrams. These Conversations do not have a special name, but are analogous to Tasks in Processes and Choreography in the way they group elements (he hexagon notation in general is technically called a Conversation Node). The Conversations on the left in Figure 8 can only group Message Flows, not other Conversations, as indicated by the absence of a small plus-sign adornment. Conversations do not capture sequencing of Message Flows, because they are the grouping element for Collaborations.

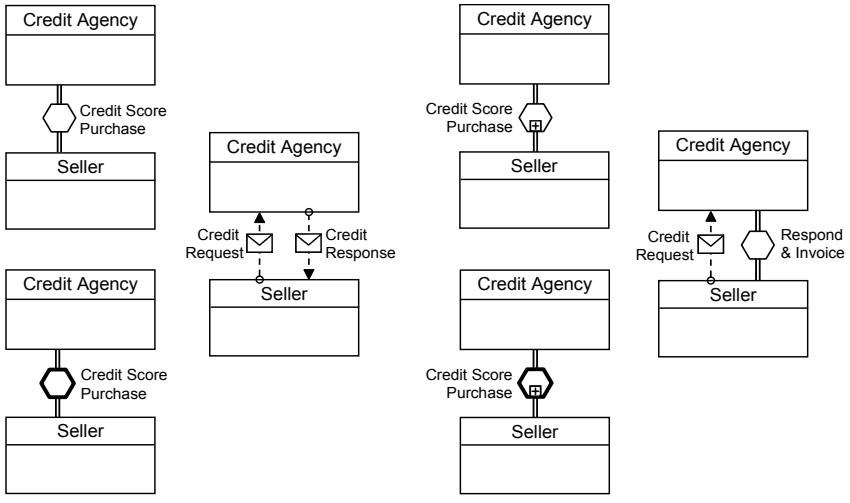

Figure 8: Conversations

To help manage complex interactions, the expansion of Conversations can include other Conversations as well as Message Flows (see examples on the right side of Figure 8). In collapsed form, these Conversations have a small plus-sign adornment. In expanded form, they appear as Message Flows and Conversations, shown on the far right (tools can show nested Collaboration diagrams separately from the Conversations they are in, to have more diagram space). The Conversations in the lower middle right is calling the Collaboration diagram on the right, which is available to be nested in multiple Activities and diagrams. The Conversation in the upper middle right is a Sub-Conversation, which nests the diagram on the right without making it available for multiple Activities or diagrams.

Collaboration diagrams sometimes call others that have different Participants, just like Choreography diagrams, see the example in Figure 9, adapted from Figure 8. The Collaboration on the right has the same Participants as Figure 8, but the Collaboration calling it on the left has Participants Retailer and Financial Services. The modeler must specify which Participants in the called Collaboration on the right should be the same as in the caller on the left. This is done with Participant Associations, as described above for Choreography, except Collaboration provides a notation. Names of the called Participants appear on Call Conversation Links to the calling Participants they are associated with. In Figure 9, Financial Services is associated with Credit Agency, and Retailer with Seller.

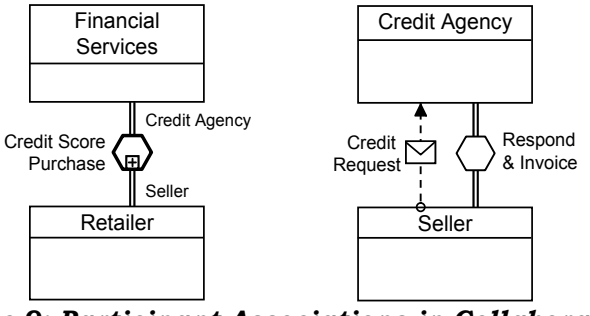

Figure 9: Participant Associations in Collaborations

Grouping Message Flows is especially useful for managing interaction complexity due to more than two Participants ("multi-party" interactions). For example, purchases using a credit card usually involve a buyer, seller, and credit card company authorizing the transaction, see the Choreography at top of Figure 10. On the top left is a Choreography Activity with three Participants, shown with an additional band. The customer initiates the interaction, as indicated by the unshaded band. The plus sign indicates expansion to a Choreography diagram, shown on the upper right. The diagram gives the details of the interaction, including Message Flow sequence and alternative flows. The Collaboration at the bottom of Figure 10 is a simplified supply chain example in which the legal receiver of the goods, the Consignee, is different from the originator of the order, the Factory, and cost is reduced by including other goods in a single shipment, as arranged by a Consolidator (adapted from an example in [5]). The original ordering Conversation is between the Factory and Supplier, while separate Conversations with the Consignee are needed for legal transfer. Messages pass between the Consignee, Consolidator, and Shipper as part of single Conversation to arrange shipment, while the Shipper agrees on a pickup location with the Supplier. High-level views like these created by grouping Message Flows are essential for modeling of complex industrial interactions.

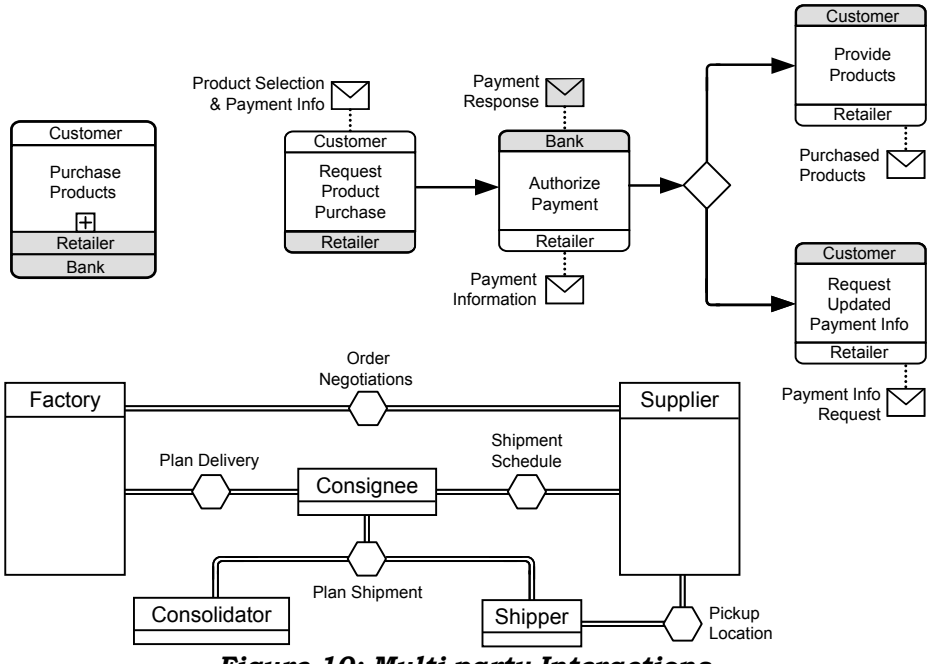

Figure 10: Multi-party Interactions

Conversations can group Message Flows in Choreographies, though they are not shown graphically in Choreography diagrams. This is typically to specify correlation information for groups of Message Flows spread across multiple Choreography Activities, see Section 4.3, but might be for other reasons, such as grouping Message Flows with the same Participants or with related Message information. Also the same Message Flow can be in multiple Conversations, see example in Section 4.3, but not in multiple Choreography Activities, because Activities are sequential in time, while Conversations are not. When Choreographies are shown inside Collaboration diagrams, Con-

versations in the Choreographies and Collaborations are linked together with Conversation Associations. Conversation Associations do not have notation, but are accessible in other ways with tools supporting interactions.

Conversations in a Choreography can be shown graphically by viewing Choreographies with Collaboration notation, using BPMN 2 support for multiple visualizations of the same content, see footnote 7. This enables display of Participants, Message Flows, and hidden Conversations of Choreographies, but in a Collaborational style that omits Sequence Flows. For example, the Choreography at the top right of Figure 10 can be visualized with Customer, Retailer, and Bank as large Participant rectangles of Collaboration diagrams, with Conversation hexagons and dashed Message Flow lines between them, as shown in Figure 11. The Conversation at the top left was hidden in Figure 10, and contains the Messages flows from the first Choreography Activity and the one on the lower right. The Conversation on the right contains the Message Flows in the second Choreography Activity.

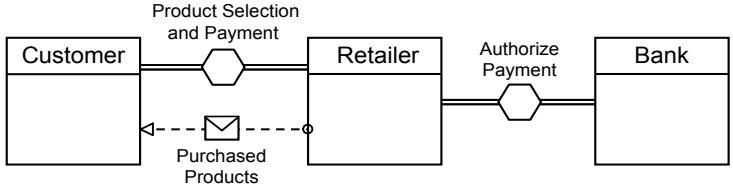

Figure 11: Collaboration Notation for Choreography in Figure 10

4. INTERACTIVE PROCESSES

Interactive Processes send and receive Messages to and from outside Participants. They are needed to deploy the interaction diagrams discussed in the previous section. Section 4.1 describes how Collaboration diagrams include interactive Processes within Participants. Section 4.2 covers interactive Processes as they appear to other Participants and to the Participants carrying them out. Section 4.3 addresses coordination of Messages with multiple ongoing Processes in Participants.

4.1 Processes in Collaboration Diagrams

Collaboration diagrams can show how Activities in a Process interact with other Participants, by sending and receiving Messages.[9] Processes usually interact with multiple Participants, serving or producing a product for at least one of them, with assistance from the others. Figure 12 is an example Process where a service is provided to a Customer, with assistance from a Credit Agency. Message Flows show which Tasks interact with which Participants (tools can give access to this information in some other way than diagrams, to save diagram space). If a Process interacts with only one Participant, a Collaboration diagram is not needed, except to specify correlation information, see Section 4.3.

[9] The same Process can appear in multiple Collaboration diagrams, but one of them is identified as the *definitional* Collaboration containing all interaction information for the Process. Definitional Collaborations show Processes in only one of their Participants, the one containing the Process being defined. Choreography diagrams cannot show Processes in Participants.

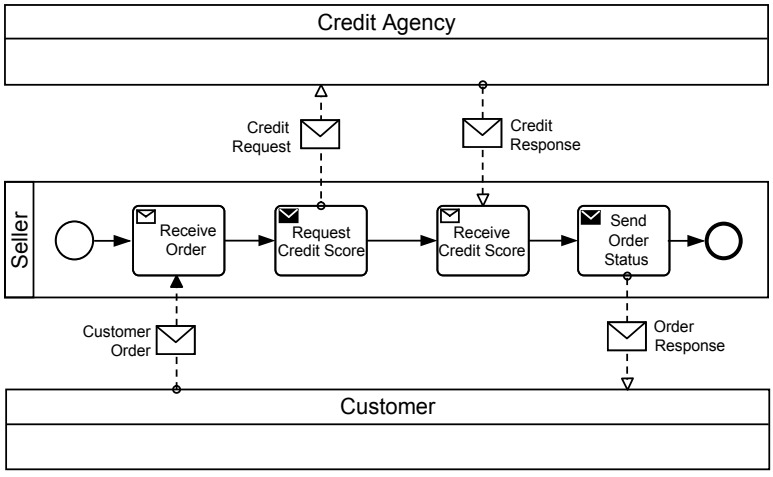

Figure 12: Process in a Collaboration

Conversations can link Process Activities and Participants by grouping many Message Flows at lower levels of Process nesting. For example, the Invoicing Conversation in Figure 13 links the Process Invoice Activity to the Invoicer Participant. The Activity calls another Process that exchanges potentially many Messages with the Invoicer (not shown for brevity). Similarly the Scheduling and Shipping Conversations might group many Message Flows to many Activities, but in this case they simplify the diagram by linking directly to the Process Participant, rather than identifying the Messages and Activities involved (Activities and Messages can be identified in the underlying model even though the diagram does not show the Message Flows or links from Conversations to the Activities). Conversations can carry information for coordinating Messages with multiple ongoing Processes in Participants, see Section 4.3.

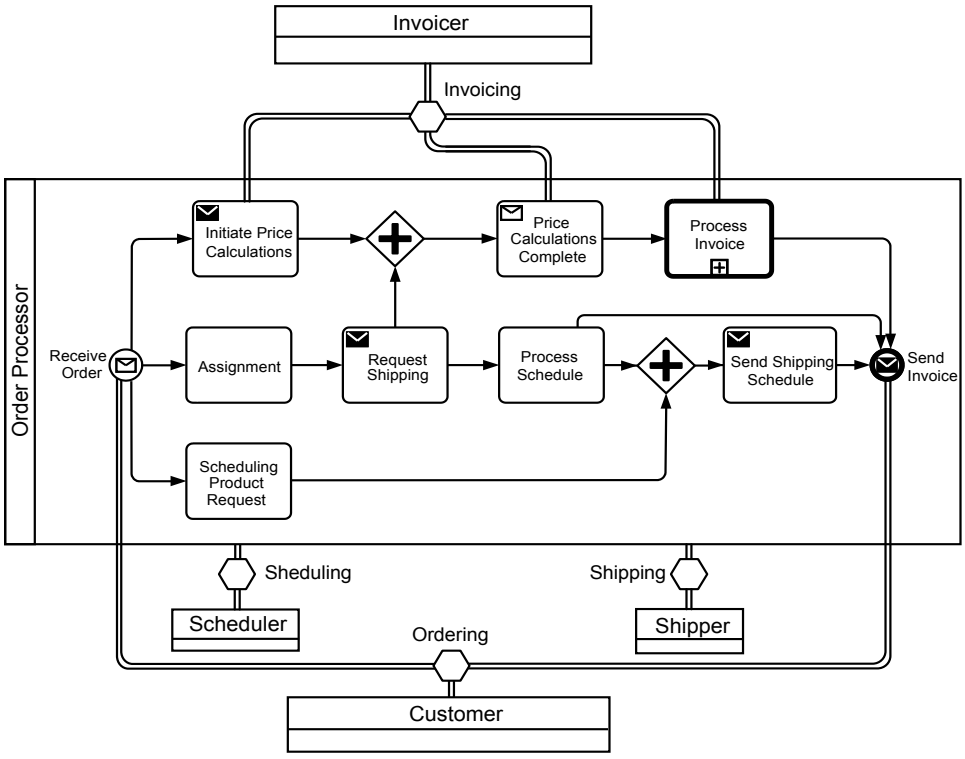

Figure 13: Process with Conversations

Interactive Processes can be nested in other Processes, and have Participants of their own. The nested Process in Figure 14 is the one from Figure 12. The outer Process has Participants Retailer and Buyer, which are linked by Participant Associations to the Seller and Customer Participants respectively in Figure 12, see Section 3.3. The Credit Agency Participant is not linked to any in the outer Process. The inner Process is responsible for choosing the particular credit agency to interact with when then Process is carried out.

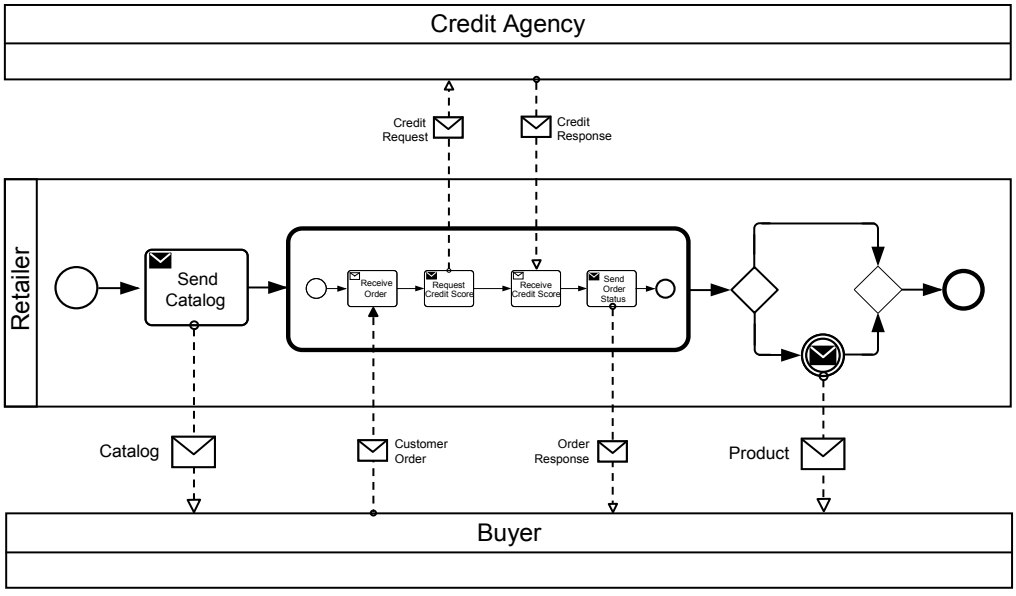

Figure 14: Nesting Interactive Processes

4.2 Public and Private Processes

Businesses usually do not show each other their Process models, even interactive ones, unless the models only describe interactions with other businesses. In BPMN 2, these are called public Processes, while those internal to a business are private.[10] For example, Figure 12 could be a public Process, because it only sends and receives Messages, which are Activities other Participants can detect. Figure 15 shows a possible private Process for Figure 12 containing Activities that are not interactive and usually not shown to other Participants. It uses Event notation for receiving and sending Messages, which has the same effect as the Receive and Send Tasks in Figure 12. While the private Process waits for the Credit Response Message, it searches internal records for credit information. A Parallel Gateway is used to receive the Credit Response Message even if it arrives while the internal search is underway. When the Credit Response Message arrives and the internal search is completed, the Process can continue. When the private Process is carried out it will appear to external Participants as if it were the public one in Figure 12, because the same Messages are sent in the same order to the same Participants.

[10] Public Processes are called "abstract" in BPMN 1.x. The name is changed in BPMN 2 to avoid confusion with Processes that act as templates for further development. Public Processes might be used as templates for private Processes, but the private Processes cannot be changed arbitrarily as those based on templates can. Private Processes must support interactions specified in public Processes (see rest of this section). Private Processes in BPMN 2 are divided into executable and non-executable, where executable Processes are fully automatable.

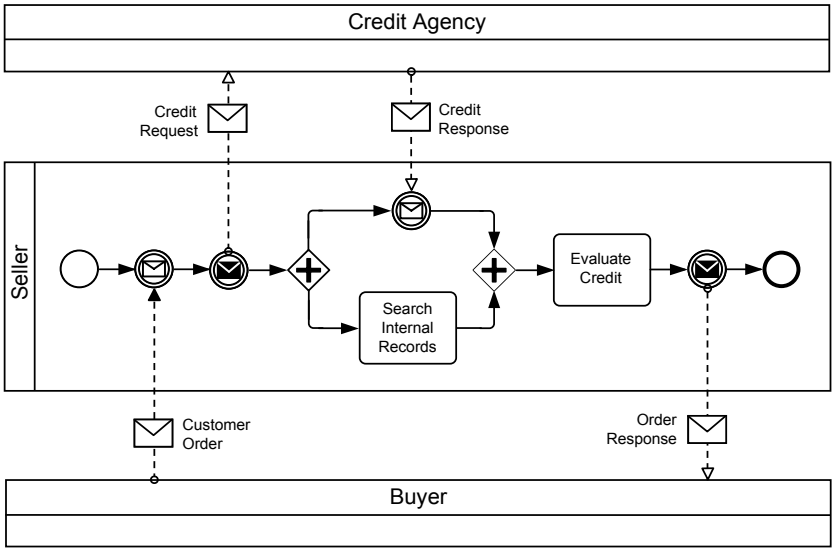

Figure 15: Private Process for Figure 12

A private Process supports a public one by interacting the same way as a public one, as described above. The *supports* relationship does not have a notation, but is accessible in other ways with tools supporting Process diagrams. Modelers can specify which private Processes support which public ones, for example, to declare a private Process they developed will cover the public Processes agreed to with partners. Tools might check these declarations, but it is not required for BPMN 2 compliance. The same private Process can support multiple public ones, each showing different aspects of the private Process. For example, two other public Processes can be defined from Figure 12 by removing either Customer or Credit Agency. These would only have Message Flows and Tasks for interacting with the remaining Participant, Credit Agency or Customer respectively. The two additional diagrams would be suitable for showing to those particular Participants. The private Process in Figure 15 would support both of these public Processes (the two public Processes would need to be open, see next paragraph).[11]

Public Processes can look like underspecified private ones. Anything not specified in public Processes is specified by private ones. For example, if none of the outgoing Sequence Flows of an Exclusive Gateway have conditions, private Processes can determine which one of the Activities immediately after the Gateway will happen. An Exclusive Gateway used this way only requires exactly one Activity immediately after it to happen, without specifying which one (the same applies if only some of the public Sequence Flows have conditions, but the conditions happen to all be false when the Process is carried out. The private Process determines which of the remaining unconditioned Sequence Flows is used). Another example is timer Events with no time specified. Private Processes can determine when the timer goes off.

[11] Definitional Collaborations typically contain private Processes for this reason, see footnote 9 in Section 4.1. All external Participants are present in a definitional Collaboration to capture all interactions, rather than only some of them for external publication. For example, customers would not see the same public Processes shown to suppliers, and vice versa.

Closed interaction diagrams prevent Processes from sending or receiving Messages other than those specified in the diagram, see Section 3.1. For example, if the Collaboration in Figure 12 is closed, then the Process in the figure and all private Processes supporting it would only be allowed to send or receive Messages already in the figure, at least when carrying out that particular interaction. Open Choreography diagrams inside Collaborations can prevent Processes from sending or receiving Messages during the time between sequential Message Flows using immediate Sequence Flows, see Section 3.2.

Public Processes can restrict whether private Processes supporting them can have additional interactive elements, even if Collaboration diagrams containing them are open. Closed public Processes prevent supporting private Processes from sending or receiving Messages other than those specified publically. For example, if the Process in Figure 12 is closed, then the private Processes supporting it could not send or receive any additional Messages. Open public Processes can prevent sending or receiving Messages during the time between particular sequential Activities using immediate Sequence Flows, similarly to immediate flows in interaction diagrams, see Section 3.2. These Sequence Flows prevent unmodeled Activities from occurring during the time between Process Activities, including interactive ones, unless other Activities are explicitly specified on parallel paths. Immediate Sequence Flows do not require the time between Activities to be zero, they only prevent unspecified Activities from occurring when the Process is carried out, including interactive ones. Process diagrams do not visually show which Sequence Flows are immediate, but this information is accessible in other ways with tools supporting Process diagrams.

Private Processes can support other private Processes, as well as public ones. This is useful for defining Processes at a low level of detail, and specializing them in various ways in separate diagrams. Specialized Processes must behave compatibly with the Processes they support. For example, a specialized private Process could support the private one in Figure 15 by adding an Activity between the first two Events, assuming the Sequence Flow between the Events is not immediate. These Events will still happen in the order defined in Figure 15, but an additional Activity will occur before the second Event does. This ensures the specialized Process will behave in a way that is still valid for Figure 15 (private Processes can be open or closed, and use immediate Sequence Flows, just like public Processes, see above in this section). Specialized Processes can be further refined by other supporting Processes, forming taxonomies of Process diagrams. Since the supporting Processes adheres to the more general ones at each level of refinement, the most specialized diagrams in the taxonomy will still behave in ways that are valid for the most general ones.

4.3 Messages and Process Instances

Messages coming into a business might start new Processes, or be routed to existing Processes already underway. For example, the initial request for a product might start a new order Process in a business, while later incoming Messages about the same order are passed along to the *process instance* already being carried out for that order. A business might handle multiple orders at the same time, each by a different instance of the same Process, described by the same Process diagram. Incoming Messages not starting a new Process instance must be routed to the existing instance handling the order

identified in the Message. Messages sent out to customers must go to the customer whose order started the particular process instance sending the Message. These concerns apply to interactive Processes in all Participants of an interaction.

Messages and process instances must contain enough information to determine if a new process instance is needed, or if not, which existing instance will handle them. This is called *correlation* information. When Messages arrive, their correlation information is matched against the corresponding information in process instances already underway. If no matches are found, a new process instance is started based on the information in the arriving Message. For example, the correlation information for ordering a product might be a customer name and product number. Arriving Messages are expected to contain this information, and order process instances already underway are also. If there is a process instance with a customer name and product number matching the one in the arriving Message, the Message is routed to that process instance. Otherwise a new process instance is created with the customer name and product number as its correlation information (a process instance might participate in multiple interactions at once. The matching requirement for correlation information only applies to Messages sent or received as part of the same interaction). Messages sent out must also contain correlation information, to enable other Participants to route incoming Messages, and to be included in replies.

Correlation information can be augmented during an interaction, and different portions of the information used with different Participants. For example, the initial Message arriving to start an order Process might have a customer name and product number, while Messages exchanged after that also have an order number assigned by the new process instance. This enables customers to order the same product multiple times before the first order process instance is finished, starting new a process instance each time because the initial order Messages do not have the order number required to match process instances already underway. Correlation information might vary during an interaction based on the Participant sending or receiving the Message. For example, an order number might be used with customers, but other identifiers for interactions with warehouses or suppliers.

Process instances can have correlation information that is the same for the duration of the instance, or that is augmented, or changed. If it is the same or only augmented during the instance, it is called *key-based* correlation. If correlation information changes more than just by augmentation, it is *context-based* correlation. Context-based correlation enables correlation information in a process instance to change over time depending on the state of the instance. Information from incoming Messages is matched against the information in process instances as it happens to be at the time of matching. This gives process instances flexibility in handling Messages over time. Information for outgoing Messages is also drawn from process instances as it happens to be at the time the Messages are sent.

Correlation information is specified on elements that group Message Flows, applying to the Messages in the flows, either directly or indirectly through nested groupings. This includes Conversations, Choreography Activities, and interaction diagrams themselves. For example, a Conversation in Figure 13 might specify correlation information, and the flows in it would have Messages with the specified information. The entire Collaboration in Figure 13

might specify additional correlation information, whereupon the complete specification for Message Flows in the Conversation would include that as well. Nested groupings can be diagrams that are called or contained in other diagrams, and correlation specifications are aggregated across these also. For example, a Conversation in Figure 13 might call a separate Collaboration diagram, or nest it directly in a Sub-Conversation. The complete correlation specification for Message Flows in the called diagram includes specifications on the calling Conversation, and the entire Collaboration in Figure 13. Correlation specifications do not have their own notation, but are accessible in other ways with tools supporting interactions.

Conversations in Choreographies can have correlation specifications, even though Conversations do not appear graphically in Choreography diagrams, see Section 3.3. These are useful for specifying correlation on groups of Message Flows not contained by a single Choreography Activity. For example, the Conversation between Customer and Retailer in Figure 11 is a view of a hidden one in Figure 10 containing Messages flows from the first Choreography Activity and the Activity on the lower right. Correlation specified on this Conversation applies to all its Message Flows, regardless of which Choreography Activity they are in. For Choreographies appearing inside Collaboration diagrams, correlation specifications are shared between associated Conversations in the Collaboration and Choreography, see Section 3.3.

When Processes appear in Collaborations, interactive elements in the Process send and receive Messages with correlation information specified by the Collaboration (see footnote 9 in Section 4.1 about definitional Collaborations for Processes). For example, in Figure 13, the Event at the end will send a Message with correlation information specified by the Ordering Conversation at the bottom, and the entire Collaboration, if any, see above. Similarly, Messages send and received by elements nested in the Process Invoice Activity will have information specified by the Invoicing Conversation at the top.

Correlation specifications are based on Correlation Properties that have values in Messages and in process instances. For example, a Correlation Property for customer names might be specified on a Conversation, and a particular customer's name will appear as the value of the property in Messages sent during the Conversation, and in process instances the Messages are routed to. The same Correlation Properties can be specified for multiple kinds of Messages. For example, a Message for submitting a new order might use one kind of form, while the Messages after that use different kinds, but they might all include the customer name as a Correlation Property. Specifications of Correlation Properties include *retrieval expressions* indicating where to find property values in each kind of Message. For example, the customer name might be the first entry in a Message for initiating an order, but appear farther down in later Messages. The same properties can be used in multiple correlation specifications, even across Conversations. In the ordering example, all Conversations might use the customer name and product number as Correlation Properties, but some Conversations might also include an order number.

Correlation specifications group their properties into Correlation Keys. When incoming Messages are routed to process instances, the values of key properties in the Messages are checked against corresponding key property values in process instances. All the properties in a key must have values in a Message for the key to be matched against a process instance, and all the

property values in the Message key must be the same as a process instance key for the keys to match. An element that groups Message Flows might have multiple keys, and a Message must have values for all properties in at least one of the keys to be included in that grouping. For example, in Figure 16 the Conversations all have one key, shown by their names in parentheses with the Conversation name (adapted from the correlation example in [6]). Since the Conversations have only one key each, Messages must have values for all properties in that key to be included the Conversation. The Message Flow for Order Request is in two Conversations at once, as indicated by two Conversation Links going into the same Receive Event. This means Messages sent under this flow must have complete values for both Conversation keys to be received by this Event.

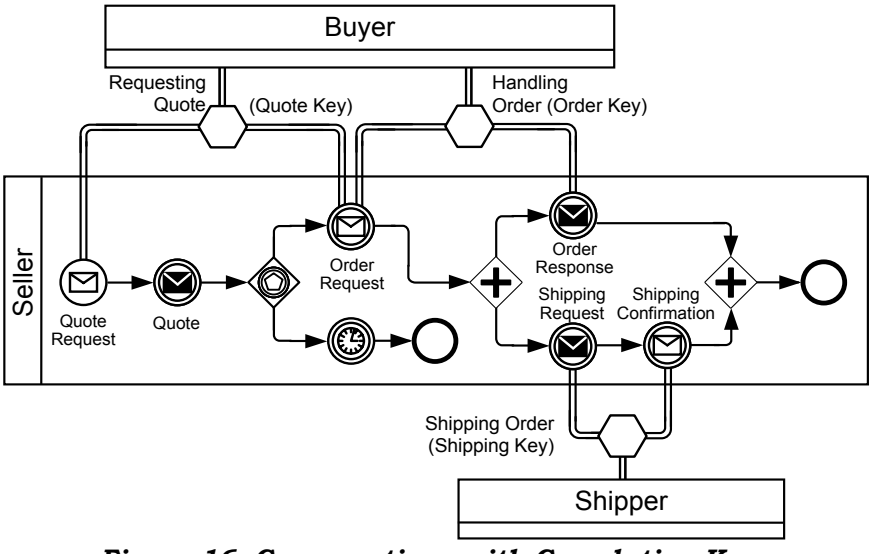

Figure 16: Conversations with Correlation Keys

Processes supporting context-based correlation calculate Correlation Property values from the current state of a Process for some or all keys used in the Process. They define Correlation Subscriptions for keys to calculate Correlation Property values dynamically, which are matched against incoming Messages.

5. CONCLUSION

Interactions and interactive Processes undergo a major upgrade in BPMN 2, with new Choreography diagrams for specifying sequences and conditions for Message Flows, Conversations grouping Message Flows in Collaboration diagrams, and explicit relationships between public and private interactive Processes. Table 1 summarizes the capabilities of the interaction diagrams in BPMN 2. Both capture Participants, Messages, and Message Flows, but Collaboration shows Participants more prominently, giving a high level view of their interaction. Choreography shows sequencing of Message Flows more directly, so is most suitable for detailed interaction modeling. Collaboration only captures Message sequencing through Processes in Participants. Collaboration is the only diagram that shows Processes, and is needed to link interactive Process Activities to external Participants. Choreography and Collaboration can group Message Flows, enabling them to scale to very complicated interactions.

	Choreography	Collaboration
Participants, Messages, Message Flow	X (Message Flow shown as Activities)	X
Sequencing Message Flow	X	
Grouping Message Flow	X	X
Processes in Participants		X
Correlating Messages and Process Instances	X (Not shown graphically)	X

Table 1: Interaction Diagram Summary

Interactive Processes are needed to deploy interactions. They can be defined to show only the public portions needed for interaction, or include private aspects also. Public and private Processes can be linked to indicate which private Processes support which public ones. Public and private Processes might appear very differently on the surface, because they are only required to have the same interactions with external Participants. Processes and interactions can control whether private Processes supporting them can send and receive additional Messages. Private Processes can also support other private Processes, to enable refinement of Processes from low to high level of detail, specializing diagrams in alternative ways.

6. ACKNOWLEDGEMENTS

The authors thank Antoine Lonjon, Peter Denno, and J.D. Baker for their input to this paper.

Commercial equipment and materials might be identified to adequately specify certain procedures. In no case does such identification imply recommendation or endorsement by the U.S. National Institute of Standards and Technology, nor does it imply that the materials or equipment identified are necessarily the best available for the purpose.

7. REFERENCES

[1] Object Management Group, “Business Process Model and Notation (BPMN), Version 2.0” http://www.omg.org/spec/BPMN/2.0, January 2011.

[2] White, S., Bock, C., “New Capabilities for Process Modeling in BPMN 2.0,” in BPMN 2.0 Handbook, 2nd ed., 2011.

[3] Organization for the Advancement of Structured Information Standards, “Web Services Coordination (WS-Coordination),” http://docs.oasis-open.org/ws-tx/wstx-wsat-1.2-spec-os/wstx-wsat-1.2-spec-os.html, January 2009.

[4] Decker, G., Barros, A., Kraft, F., Lohmann, N., “Non-desynchronizable Service Choreographies,” in Proceedings of the Sixth International Conference on Service-Oriented Computing, Springer, Notes in Computer Science, Vol. 5364, pp. 331-346, 2008.

[5] Barros, A., Decker, G. and Dumas, M., “Multi-staged and Multi-viewpoint Service Choreography Modelling,” in Proceedings of the Workshop on Software Engineering Methods for Service Oriented Architecture (SEMSOA), CEUR Workshop Proceedings, Vol. 244, May 2007.

[6] Object Management Group, “BPMN 2.0 by Example,” http://www.omg.org/spec/BPMN/2.0/examples/PDF/10-06-02.pdf, June 2010.