Encyclopedia of Networked and Virtual Organizations

Goran D. Putnik
University of Minho, Portugal

Maria Manuela Cunha
Polytechnic Institute of Câvado and Ave, Portugal

Volume I
A–F
Collaboration Based on Web Services

Giorgio Bruno  
*Dip. Automatica e Informatica, Politecnico di Torino, Italy*

Marcello La Rosa  
*Queensland University of Technology, Australia*

**INTRODUCTION**

Web services are software components designed to support interoperable machine-to-machine interactions over a network, through the exchange of SOAP messages. Since the underlying technology is independent of any specific programming language, Web services can be effectively used to interconnect business processes across different organizations. However, a standard way of representing such interconnections has not yet emerged and is the subject of an ongoing debate.

In this area, the term *collaboration* has often been used to denote a situation in which two or more business processes (participants) cooperate by means of Web services, so as to achieve a common goal. In particular, when only two participants are involved, the more specific term *binary collaboration* is preferred, while when there are three or more, the term *multi-party collaboration* can be used instead. The notion of binary collaboration is fundamental, as any multi-party collaboration ultimately relies on a number of binary collaborations.

Collaborations can be described by collaboration models, which provide a control-flow view of the intended global behaviour. These models are addressed from two perspectives, one focusing on the observable activities of the participants, and the other on their interactions.

The first approach defines the observable activities of the participants as well as their ordering constraints by means of a global model called inter-organizational workflow. As an example, the public-to-private approach (van der Aalst & Weske, 2001) is a top-down technique based on three steps: at first the participants agree on a global model represented by a Petri net, then the public model is partitioned into public parts, one per participant, and finally each participant refines its public part into a private workflow. The refinement process guarantees that the private workflows conform to the global model.

The approach focusing on interactions is more abstract. In fact, interactions are carried out by message-sending activities and message-receiving ones. There are two types of interactions: one-way interactions and two-way ones; a two-way interaction consists of two one-way interactions in the opposite directions. As a matter of fact, a one-way interaction subsumes two activities, a message-sending activity in one business process, and a message-receiving activity in another.

This article follows the interaction-oriented approach and illustrates binary collaborations and multi-party ones with the help of an informal notation called “interaction diagrams.”

**MODELS OF BINARY COLLABORATIONS**

Well known models of binary collaborations are the partner interface processes (PIPs) developed by the RosettaNet consortium (Damodaran, 2004). A PIP refers to two roles, consists of a one-way or two-way interaction, and specifies the business documents to be exchanged as well as the quality of service (QoS) requirements (such as timeToPerform, timeToAcknowledgeReceipt and retryCount).

As an example, PIP 3A1, “request quote,” shown in Figure 1, enables a buyer to request a product quote from a provider, and a provider to respond accordingly. The model is based on the modeling methodology promoted by UN/CEFACT (“UMM,” 2003) and referred to as UMM. The two participants are represented by a pair of business activities: getQuote is the requesting activity and provideQuote is the responding one. RequestForQuote and quote are called action messages. Action messages are acknowledged by positive or negative signal messages, not shown in Figure 1; usually a signal message acknowledges that an action message has been received and has been syntactically validated. The business activities are involved in four
message exchanges, concerning two action messages and two signal messages.

The model in Figure 1 conveys the transactional nature of the interaction: the requesting activity ends in two alternative states, “end” and “failed.” The success state (end) indicates that all the messages have been properly received. The failure state takes into account all the possible exceptions, which can be divided into communication exceptions, business exceptions (when a message is not understood) and timeout ones.

The business transaction activities defined in UMM are similar to RosettaNet PIPs. UMM allows business transaction activities to be combined into choreographies, which are called “business collaboration protocols.” They are modelled as universal modeling language (UML) (“UML,” 2005) activity diagrams and can include four control-flow elements: decision, merge, fork, and join. An equivalent XML representation can be obtained by means of the business process specification schema (BPSS), which is part of the ebXML framework (“BPSS,” 2001).

The current version of BPSS supports binary collaborations only, while UMM addresses choreographies. In fact, UMM does not address binary collaborations specifically as the business transaction activities contained in the same business collaboration protocol may be performed by different pairs of business partners (Hofreiter, Huemer, & Kim, 2004). BPSS, unlike UMM, allows business collaborations to be nested.

The notation informally presented in this article (i.e., interaction diagrams) draws on UMM with two major differences.

The UMM notation is affected by redundancy. In fact, the business activities appearing in a UMM business transaction activity (which is similar to a RosettaNet PIP such as the one shown in Figure 1) do not play any functional role; they only serve as a support for QoS parameters. Therefore, the major building blocks in interaction diagrams are the interactions, which do not need to be further decomposed but can directly be associated with QoS parameters. Collaboration “RFQ” shown in Figure 2a is the equivalent of PIP 3A1 presented in Figure 1. It is a simple binary collaboration consisting of a single two-way interaction.

In an interaction diagram, the two participants involved in the binary collaboration are denoted by their roles such as buyer and supplier; roles appear in brackets after the collaboration name. The first role designates the collaboration initiator (or requester), and the second one designates the collaboration provider.

Interactions are depicted as rectangles containing the names of the messages involved and can be labelled with identifiers, such as i1. In two-way interactions, a slash (/) separates the request message from the response one. The types of the messages are defined in an XML schema file associated with the collaboration model.

An interaction takes place between two participants, denoted by two conventional roles, for example, initiator (or requester) and responder (or provider). The collaboration requester coincides with the initiator of the first interaction. If an interaction is initiated by the collaboration provider, the request message is underlined (this case does not appear in Figure 2).

The second difference between interaction diagrams and UMM lies in the way of expressing conditions in control-flow elements: a condition in UMM is mainly related to the availability of a business object in a given state. However, collaborations should not depend on external entities, such as the business objects postulated in UMM conditions, as different interpretations can be
associated with them by the parties involved. Instead, conditions must depend on information agreed on by the parties, such as the content of messages.

Binary collaboration “RfQ-Order,” shown in Figure 2b, presents a sequence of two interactions, where the second one is optional as the buyer sends the order only if the quote is satisfactory. Optional interactions are based on the deadlines associated with interactions. If a deadline expires, a timeout (represented by a dashed link) occurs indicating that the interaction did not take place in due time. If there is no timeout link, the interaction fails and the whole collaboration fails; otherwise, the interaction is optional and the timeout link shows the next step to be undertaken. The rfq is assumed to include deadline to: if the order is not sent/received before to, the collaboration will be ended, hence the timeout link (i.e., the dashed link) connects interaction i2 to the final state. The script shown in the “deadlines” section (i.e., “i2.d = rfq.to” sets the deadline (denoted by d) of interaction i2 to the value read from attribute to of rfq). Past messages act as global variables for such scripts.

Binary collaboration “RfQ-Order1,” shown in Figure 2c, presents a more complex protocol, in which a quote is assumed to include a Boolean attribute (negotiable) indicating whether it is negotiable or not. After receiving a quote, the buyer can send an order or a revised request for quote, if the quote is negotiable. In the latter case, two alternative paths are possible, one consisting of interaction order and the other starting with interaction rfq/quote.

Two state-based patterns (van der Aalst, ter Hofstede, Kiepuszewski, & Barros, 2003) appear in Figure 2c: state s1 enables a data-driven exclusive choice, while state s2 represents an event-driven (or deferred) choice. In the first case, the choice must be based on public information, visible to both parties: such public information is given by the contents of past messages (i.e., the messages exchanged by the parties before the choice is made). In the second case, the choice depends on the arrival of future messages.

When collaboration RfQ-Order1 is in state s1, a data-driven choice takes place, depending on the contents of the last quote. The conditions appearing on outgoing links determine which link has to be followed.

State s2 determines an event-driven choice. An event-driven choice occurs when a place is followed by two or more interactions in the same direction. The collaboration remains in state s2 until an interaction occurs or deadline s2.d expires; in the latter case, the collaboration will be ended, as shown by the timeout link.

In addition to states, interaction diagrams can include fork elements and join elements as shown in the next section.

CHOREOGRAPHIES

Choreography denotes an a priori global model meant to capture all the interactions taking place for a given purpose among a number of participants. As such, it is
a much debated notion. It is often associated with the idea of a leading organization having the authority of imposing the required behaviour on the participating organizations. Three points of weakness have been pointed out: a leading organization may not exist, a participant may be willing to select its own partners, and participants are exposed to unnecessary information (Zhao, Liu, & Yang, 2005).

There are cases, however, in which binary collaborations are not sufficient to define the mutual behavior of the participants; this happens not because the participants are more than two, but because the order implied by binary collaborations, although necessary, is not sufficient to describe how interactions are expected to take place. In such cases, choreographies are needed.

As an example, a supply chain involving a buyer, a distributor, and a supplier is considered, as follows.

A buyer sends a purchase order \((bo = buyer\ order)\) for certain goods to a distributor which fulfils the order with two deliveries, an internal delivery \((dd = \text{distributor delivery})\) coming from an internal warehouse, and an external delivery \((sd = \text{supplier delivery})\) coming from an external supplier. The distributor sends a purchase order \((do = \text{distributor order})\) to the supplier which replies with a confirmation \((sc = \text{supplier confirmation})\), and then it informs the buyer of the supplier involved with a notification message \((dn = \text{distributor notification})\). The buyer sends some delivery information \((bi = \text{buyer information})\) to the supplier. After the deliveries \((dd\ and\ sd)\) have been performed, the buyer makes a payment in favor of the distributor and sends it a payment notification \((bp = \text{buyer payment})\). After delivering the goods to the buyer, the supplier sends a payment request \((sr = \text{supplier request})\) to the distributor; after making the payment in favor of the supplier, the distributor sends it a payment notification \((dp = \text{distributor payment})\).

A business case is informally illustrated in the sequence diagram presented in Figure 3a; it shows that interaction \(do/sc\) is nested in interaction \(bo/dn\).

This case study presents the routing pattern called “request with referral” (Barros, Dumas, & ter Hofstede, 2005), as the distributor sends a purchase order to the supplier and then the supplier sends a delivery message to the buyer indicated in the order.

The choreography related to the case study (choreography “BDS”) is shown in Figure 3b along with the binary collaborations involved (Figure 3c), all the diagrams being interaction diagrams. Binary collabora-

---

**Figure 3. Models related to the supply-chain case study**

(a) Business case

(b) Choreography BDS

(c) Binary collaborations
Collaboration Based on Web Services

tions are named after the initials of the roles involved; hence bd is the binary collaboration between the buyer and the distributor.

An extended notation is used in the choreography, as each interaction is preceded by the identifier of the binary collaboration it belongs to.

There are two two-way interactions (i.e., bo/dn and do/sc); the others are one-way interactions. All the interactions are mandatory. Three binary collaborations are implied (i.e., bd, ds and bd). They are identified by the initials of the participants, the first letter denoting the initiator.

Choreography BDS includes four control-flow activities (i.e., two forks and two joins). Fork f1 is needed because the two deliveries may take place in any order. Fork f2 enables both join j1 and the request for payment from the supplier. Join j1 is needed because the payment in favor of the distributor is made after the two deliveries. The bold link from bo/dn to do/sc is the nesting operator: its source is a two-way interaction and its destination is a nested interaction or a nested choreography.

While choreography BDS includes all the interactions appearing in the binary collaborations, this is not always the case. In fact, only the interactions needed for global coordination must be shown in the choreography, while those related to the details of the binary protocols only appear in the binary collaborations. This is the reason why, in general, choreographies cannot replace binary collaborations. As an example, interaction ds.dp can be dropped from the choreography, as it follows interaction ds.sr (which is included in the choreography) on the basis of binary collaboration ds.

Binary collaborations are needed as they drive the implementation; a previous paper has shown how to generate BPEL (business process execution language) (Andrews et al., 2003) processes from binary collaborations (Bruno & La Rosa, 2006).

CONCLUSION

This article has informally presented a modeling notation called interaction diagrams, that allows collaboration models, both binary models and multi-party ones, to be homogeneously represented.

Despite the existence of a notation for the high-level modeling of business processes (i.e., the business process modeling notation (BPMN) (White, 2004)—a de-facto standard that can also be used to represent inter-organizational workflows—there is no widely accepted notation for interaction models. A recent proposal presents an interaction-oriented language, “Let’s Dance” (Zaha, Dumas, ter Hofstede, Barros, & Decker, 2006), which, instead of using the standard control-flow elements (decisions, forks, and joins), organizes the interactions by means of three binary relationships—precedence, weak precedence, and inhibition.

While binary collaborations are well understood, in choreographies there are still several issues to be settled. For example, a set of well-formedness rules for regulating message dependencies has not been uniformly defined so far. Moreover, the mapping from choreographies to abstract orchestration models (i.e., models describing how a given participant has to deal with the collaborations it is involved in), still lacks a proper formalization. In fact, while collaboration models establish how the parties have to interact so as to achieve a common goal, it is up to each party to orchestrate (i.e., to combine) the collaborations it is involved in.

In this field, a recent effort to describe choreographies by means of an XML-based interaction-oriented language has yielded the Web services choreography description language (WS-CDL) (Kavantzas, Burdett, & Ritzinger, 2004). However, its complexity and the lack of a proper notation have raised numerous doubts (Barros, Dumas, & Oaks, 2005).

Future research should be aimed at addressing the previously mentioned shortcomings of collaboration modeling.

REFERENCES


KEY TERMS

Abstract Orchestration Model: An abstract business process made up of communication activities and control-flow ones, describing how a given participant deals with the collaborations it is involved in.

Binary Collaboration: A collaboration involving only two participants.

Choreography: An a priori global and public model meant to capture all the interactions taking place for a given purpose among a number of participants.

Collaboration: A composition of interactions whereby two or more participants exchange messages in order to achieve a common goal.

Collaboration Initiator or Requester: The participant sending the first message in a collaboration. This role coincides with the initiator of the first interaction.

Collaboration Provider: The participant involved in a collaboration without having initiated it.

Interaction Initiator or Requester: The participant sending the first message of an interaction.

Interaction Responder or Provider: The participant involved in an interaction without having initiated it.

Interaction: The exchange of a given application message between two participants. A one-way interaction subsumes two activities, a sending activity in one participant, and a receiving activity in the other. Two-way interactions are made up of two one-way

Collaboration Based on Web Services
Collaboration Based on Web Services

interactions in opposite directions. Application messages can be acknowledged by positive or negative signal messages.

**Multi-Party Collaboration:** A collaboration involving more than two participants. Any multi-party collaboration ultimately relies on a number of binary collaborations.