On Collaborations and Choreographies

Giorgio Bruno[1], Giulia Bruno[1], Marcello La Rosa[2]

[1] Dip. Automatica e Informatica, Politecnico di Torino,
24 Corso Duca degli Abruzzi, 10129 Torino, Italy

[2] Centre for Information Technology Innovation,
Queensland University of Technology,
GPO Box 2434, Brisbane QLD 4001, Australia

Abstract. This paper analyzes binary collaborations and multi-party collaborations in the context of business processes and proposes a lifecycle in which collaborations are first represented with abstract models called collaboration processes, then embodied in business processes and finally implemented in BPEL. In particular this paper discusses how to represent multi-party collaborations and presents two approaches: one is based on binary collaborations complemented with choreographies, and the other draws upon the notion of extended binary collaborations.

1 Introduction

Collaborative business processes are meant to collaborate with each other so as to achieve a common goal. Therefore great importance is attached to the various notions of collaborations, i.e. binary collaborations and multi-party ones. In this context, "party" subsumes "business process" in that the parties are the organizations responsible for the collaborative business processes.

A binary collaboration mainly refers to the sequence of interactions taking place between two parties for a specific purpose over a given period of time. An interaction can be based on a single message (asynchronous interaction) or on a pair of messages in the opposite directions (synchronous interaction). In an asynchronous interaction one party (the initiator of the interaction) sends a message to the other party (the follower), which upon receiving the message will perform some processing, i.e. an operation, without the initiator being affected in any way. In a synchronous interaction the initiator sends a "request" message, then waits for a "response" message from the follower; the follower is meant to perform an operation upon receiving the request message and to send a result back to the initiator with the response message.

Collaborations imply a lifecycle, as their development proceeds through a number of phases. At specification time, the parties involved have to agree on the interactions and the order in which they are to be carried out; an abstract model is needed and we refer to it as a collaboration process. At design time the parties work out their

Bruno G., Bruno G. and La Rosa M. (2006).
On Collaborations and Choreographies.
In *Proceedings of the 1st International Workshop on Technologies for Collaborative Business Process Management*, pages 3-12

business processes so as to conform to the intended collaborations; collaborations are embodied in communication activities. At run time collaborations are mapped to mechanisms suitable for correlating the actual messages to the actual process instances; BPEL [1] has been adopted as the implementation language.

During this research an environment, called bProgress [2], has been developed: it consists of a number of tools allowing users to manage the lifecycle of collaborations. In particular, bProgress includes a translator which is able to automatically generate BPEL processes from collaboration models and business process models.

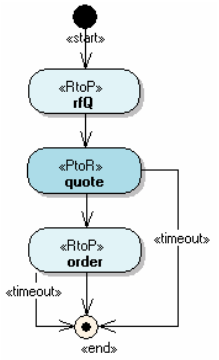
A multi-party collaboration can be still thought of as the combination of a number of binary collaborations, which, in general, are not independent of each other. Usually multi-party collaborations are described by means of choreographies, whose aim is to encompass all the relevant interactions from a global perspective and to specify precedence and timing constraints. Choreographies are not meant to completely replace binary collaborations, as not all the interactions concern all the parties; therefore choreographies can be interpreted as global constraints imposed on binary collaborations.

This paper discusses the relationships between binary collaborations and choreographies and is organized as follows. Section 2 presents how binary collaborations and collaborative business processes are represented in bProgress. Section 3 addresses multi-party collaborations and presents two approaches: the first approach is based on choreographies, the second one on extended binary collaborations (without choreographies). Section 4 presents the conclusion.

2 Binary Collaborations

An example of binary collaboration is the one established between a buyer and a seller, as follows. The party which starts the collaboration is called the requester; the other is called the provider. The requester (i.e. the buyer) starts a new collaboration by sending a request for quote, which includes the description of the goods (or services) needed, the identifier of the requester (requesterId) and two deadlines, tQ and tO. The provider can then send a quote and the requester will wait for it, but if the quote is not sent before tQ, the collaboration will be ended. A quote includes the identifier of the provider (supplierId) and the cost of the goods. After receiving a quote, the requester can send a purchase order and the provider will wait for it, but if the order is not sent before tO, the collaboration will be ended.

The model of this collaboration, referred to as bsC (i.e. buyer-seller collaboration), is the collaboration process shown in Fig. 1. A collaboration process is a number of interactions placed within a control structure providing for sequential, alternative, and timeout-related paths. It is represented in bProgress as a special activity diagram, for which we have developed a particular UML profile [3], called collaboration profile (not shown due to space limitations).



Attributes
- rfQ: type = *rfQType*
- quote: type = *quoteType*, deadline = *rfQ.tQ*
- order: type = *orderType*, deadline = *rfQ.tO*

Schema bsCX.xsd
- rfQType: string *collaborationId*,
  string *requesterId*,
  string *description*, dateTime *tQ*,
  dateTime *tO*
- quoteType: string *collaborationId*,
  string *supplierId*, double *cost*
- orderType: string *collaborationId*,
  string *requesterId*, double *cost*

Fig. 1. The model of collaboration bsC.

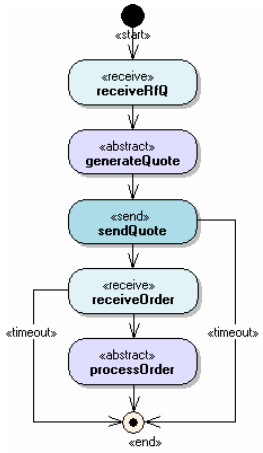
The "RtoP" stereotype indicates an asynchronous interaction whose initiator is the requester, while the "PtoR" stereotype indicates an asynchronous interaction whose initiator is the provider. Synchronous interactions are denoted by stereotypes "sRtoP" and "sPtoR". The types of the messages are provided in a schema file associated with the collaboration model.

A collaboration model implies the existence of two business processes, one on the requester's side and the other on the provider's side, whose instances are meant to carry out actual collaborations conforming to that model. An actual collaboration is a sort of logical link between two process instances and how this link is implemented depends on the underlying platform. The BPEL run time system uses a technique called correlation so as to deliver the incoming messages to the appropriate process instances. The matching between an incoming message and a process instance waiting for a message of that type is based on particular attributes of the message, called properties. The properties of messages are meant to be agreed upon by the parties involved in the collaboration; hence they are logically part of the collaboration model. In bProgress we adopted a convention that enables the translator to generate BPEL code in a standard way. In fact we assume that each message contains a particular attribute, called collaborationId, as shown in Fig. 1: its value is set by the requester before starting a given collaboration and then it is used in all the subsequent messages within that collaboration. A suitable collaborationId can be the URL of the requester process concatenated with the identifier of the requester process instance and with an integer value that the requester increments before starting a new collaboration.

Two simplified business processes which belong to different sellers and provide the same bsC collaboration are presented in Fig. 2. Process salesBP1 is basically an extension of bsC, in which "RtoP" interactions have been turned into "receive" activities and "PtoR" interactions have been turned into "send" activities. We think of business processes as extended UML 2.0 activity diagrams, and for this purpose we have developed a specific UML profile, called process profile. In our approach a business process is meant to be automatically translated into a BPEL process, therefore those stereotypes (and their attributes) aim at facilitating such translation.

A business process keeps local information in its (process) variables: they do not need to be explicitly declared, as they can be taken from the inVar and outVar attributes of the process activities.

Communication activities, i.e. the ones denoted by stereotypes “send” or “receive”, indicate the collaboration model and the interaction they refer to, by means of attribute interaction. Attribute inVar indicates into which variable the input message is to be copied, and attribute outVar indicates from which variable the output message is to be taken. Process salesBP1 is simplified in the sense that actual processing activities are ignored; however stereotypes “abstract” act as place cards for them. For this reason salesBP1 can also be interpreted as a behavioral interface model [4].



Attributes

- receiveRfQ: interaction = *bsC.rfQ*, inVar = *rfQ*
- generateQuote: outVar = *rfQ*, inVar = *quote*
- sendQuote: interaction = *bsC.quote*, outVar = *quote,* deadline = *rfQ.tQ*
- receiveOrder: interaction = *bsC.order*, inVar = *order,* deadline = *rfQ.tO*
- processOrder: outVar = *order*, inVar = *ack*

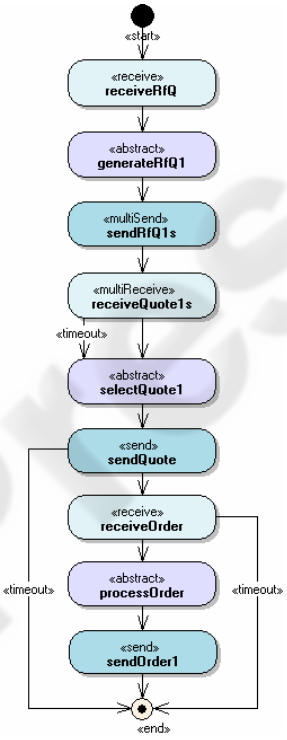


Fig. 2. The models of business processes salesBP1 (left) and salesBP2 (right).

Process salesBP2 is more complex in that it acts as a broker. In fact, it relays the request for quote to a number of suppliers, and then selects the best of the quotes received and relays it to the buyer; if it receives the order it will relay it to the supplier selected. For simplicity’s sake the attributes of the activities have been omitted.

The collaborations between salesBP2 and its suppliers are similar to the one existing between the buyer and salesBP2; those collaborations (referred to as multiple collaborations) can be handled collectively by means of two new activity types called “multiSend” and “multiReceive”. These activities carry out patterns “one-to-many send” and “one-from-many receive” illustrated in [5]. A multiSend activity, such as

sendRfQ1s, broadcasts the same message to all the parties involved in the multiple collaborations, whilst a multiReceive activity, such as receiveQuote1s, is meant to receive a number of similar messages (i.e. a message from each of the parties involved).

Although similar, the collaboration between the buyer and the broker and the one between the broker and any particular supplier are independent of each other (as they are unaware of each other) and hence they do not form what we call a multi-party collaboration, the topic of the next section.

3 Multi-party Collaborations

Generally speaking, a multi-party collaboration takes place when three or more parties are involved in (binary) collaborations that are not independent of each other, in the sense that there is a global awareness of the relevant interactions. The case study that follows is concerned with a business protocol to be established among a number of organizations acting as customers, brokers or suppliers. We are interested in the second part of that protocol, i.e. from orders to payments, whilst the first part deals with requests for quotes and quotes, as shown in the previous section.

A customer sends a purchase order (orderCB) for certain goods to a broker which sends a third-party order (orderBS) to a supplier. As soon as the supplier has completed the delivery to the customer, it sends a delivery document to the customer. The customer then sends an acceptance document to the broker. On the basis of their respective agreements the customer sends a payment to the broker (paymentCB) and the broker sends a payment (paymentBS) to the supplier. The deadlines of the interactions are as follows: the delivery of goods has to take place within the delivery date specified in the order, the acceptance will be sent within 3 days from the delivery date, paymentCB will be made within 30 days from the acceptance date, and paymentBS will be made within 40 days from the acceptance date. For simplicity's sake only the case of successful acceptance will be considered. The above mentioned interactions can be illustrated, as an UML sequence diagram, in the global model shown in Fig. 3.

The three binary collaborations implied by the global model are shown in Fig. 4. The binary collaboration (cbC) between the customer and the broker consists of interactions orderCB, acceptance and paymentCB; however acceptance is not a direct consequence of orderCB even if it is preceded by orderCB. In fact, message acceptance will be sent only after message delivery has been received. Therefore the link from orderCB to acceptance is dashed as it implies weak precedence; likewise for the link from orderBS to paymentBS. If the order of the interactions within a given collaboration is not entirely determined by these interactions, the collaboration is not self-contained. Of the three collaborations illustrated in Fig.4, only the one between the supplier and the customer is self-contained, while the others are not. As a consequence, not all the deadlines of the interactions can be precisely determined.

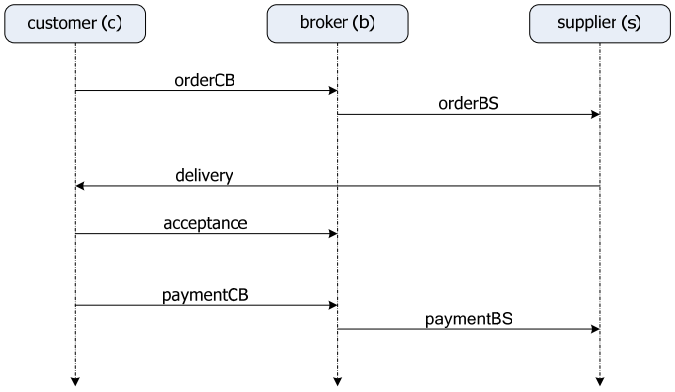
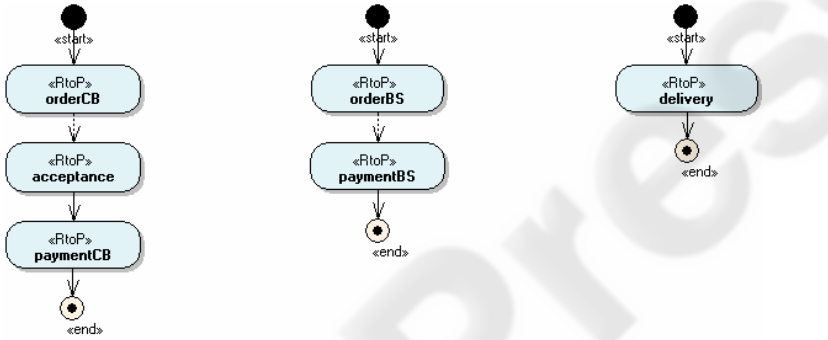


Fig. 3. The global model of interactions.



Attributes
- orderCB:
  type = *orderCBType*
- acceptance:
  type = *acceptanceType*
- paymentCB:
  type = *paymentCBType*,
  deadline = *acceptance.date+30*

Schema cbCX.xsd
- orderCBType:
  dateTime *deliveryDate*
- acceptanceType:
  dateTime *date*

Attributes
- orderBS:
  type = *orderBSType*
- paymentBS:
  type = *paymentBSType*

Schema bsCX.xsd
- orderBSType:
  dateTime *deliveryDate*

Attributes
- delivery:
  type = *deliveryType*

Schema csCX.xsd
- deliveryType:
  dateTime *date*

Fig. 4. The models of binary collaborations cbC (left), bsC (center) and csC (right).

In fact, the deadline of acceptance in collaboration cbC cannot be expressed as it depends on the date of the delivery (which is not included in the collaboration).

The problem is then how to represent a multi-party collaboration, without unnecessary information being revealed to the parties – as it would happen if a global interaction model were provided. We propose two solutions: the first is based on binary collaborations plus a choreography model, and the second is based on extended binary collaborations.

3.1 Choreographies

Binary collaborations are needed as parties interact in pairs, however if binary collaborations are not self-contained, an additional ordering structure is required. Such a structure is called choreography as it establishes precedence constraints among the interactions of different collaborations. The choreography related to the case study (cbsCh = customer-broker-supplier choreography), shown in Fig. 5, is based on a UML profile, called choreography profile, which essentially features the "interaction" stereotype and control flow constructs. Like collaboration processes, choreographies are made up of a number of interactions placed within a control structure providing for sequential, alternative, and timeout-related paths.

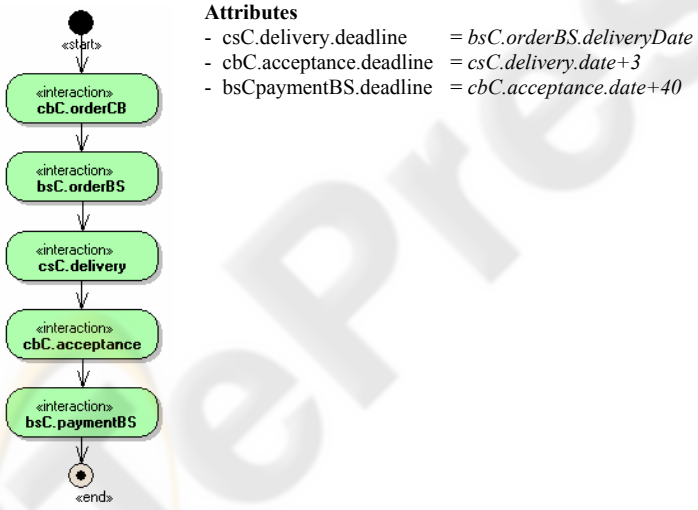
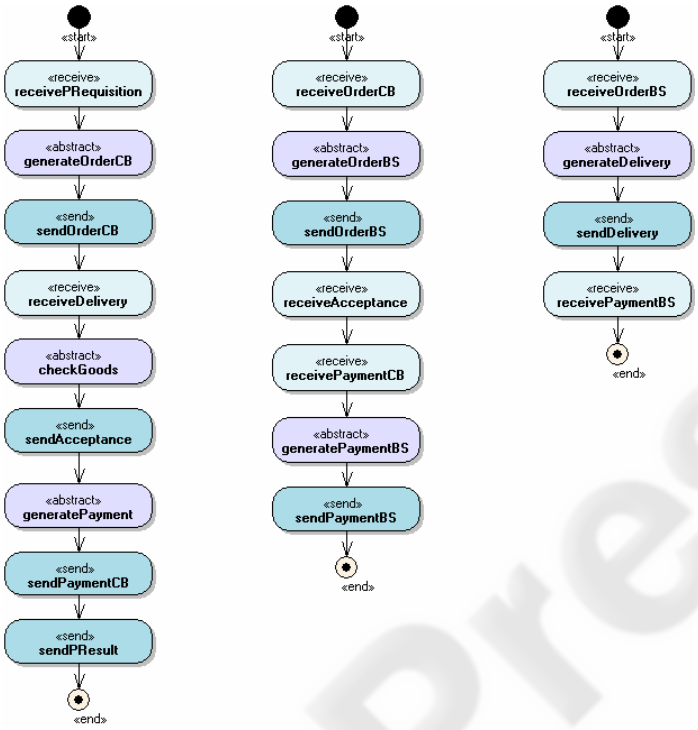


Fig. 5. Choreography cbsCh.

Basically choreography cbsCh provides the missing links to the binary collaborations shown in Fig. 4, and if we "merge" cbsCh and the binary collaborations we come up with the global model shown in Fig. 3. The business processes of the parties involved in the case study are shown in Fig 6. There are some deadlines that cannot be precisely determined, such as that of activity receiveAcceptance in process brokerBP. This happens because brokerBP does not receive message delivery on which that deadline is based. However, as a reasonable solution, that deadline could be set to the upper limit, which is given by orderCB.deliveryDate + 3. On the contrary, the

deadlines of the "send" activities can be precisely established. If we want all the parties to be able to compute the deadlines precisely, we have to add other messages to the business protocol; in particular the delivery message should be sent to the broker, too, and the acceptance message to the supplier, too.



Attributes of customerBP
- sendOrderCB: interaction = *cbC.orderCB,* outVar = *orderCB*
- receiveDelivery: interaction = *csC.delivery*, inVar = *delivery*, deadline = *orderCB.deliveryDate*
- sendAcceptance: interaction = *cbC.acceptance*, outVar = *acceptance*, deadline = *delivery.date+3*
- sendPaymentCB: interaction = *cbC.paymentCB*, outVar = *paymentCB*, deadline = *acceptance.date+30*

Attributes of brokerBP
- receiveOrderCB: interaction = *cbC.orderCB,* inVar = *orderCB*
- sendOrderBS: interaction = *bsC.orderBS*, outVar = *orderBS*
- receiveAcceptance: interaction = *cbC.acceptance,* inVar = *acceptance*, deadline = *???*
- receivePaymentCB: interaction = *cbC.paymentCB,* inVar = *paymentCB*
- sendPaymentBS: interaction = *bsC.paymentBS*, outVar = *paymentBS*, deadline = *acceptanceDate+40*

Attributes of supplierBP
- receiveOrderBS: interaction = *bsC.orderBS,* inVar = *orderBS*
- sendDelivery: interaction = *csC.delivery*, outVar = *delivery*, deadline = *orderBS.deliveryDate*
- receivePaymentBS: interaction = *bsC.paymentBS,* inVar = *paymentBS*, deadline = *???*

Fig. 6. The business processes of the customer (left), broker (center) and supplier (right).

3.2 Extended Binary Collaborations

This approach is completely decentralized in the sense that it does not require any common model. The parties, instead, working in pairs, through an iterative process, will come up with extended binary collaborations. A binary collaboration is extended if it includes extended interactions, an extended interaction taking place between one of the two parties (involved in the collaboration) and an external one. The extended binary collaborations related to the case study are informally shown in Fig. 7, as UML sequence diagrams. The collaboration between the customer and the broker is extended because it includes the delivery message from the supplier to the customer.

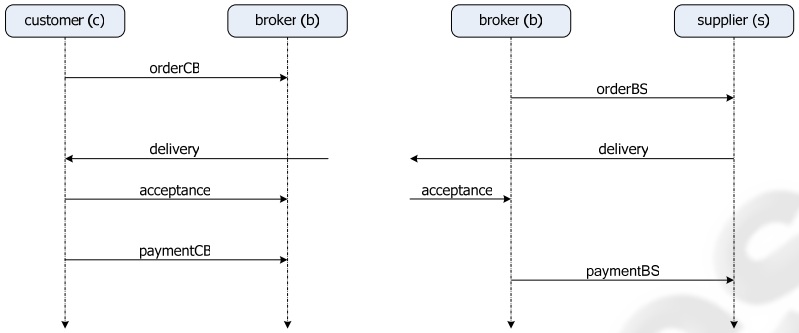


Fig. 7. Extended collaborations cbC and bsC depicted as UML sequence diagrams.

The extended collaboration models are shown in Fig. 8. Extended interactions are represented by stereotypes “toP”, “toR”, “fromP” and “fromR”.

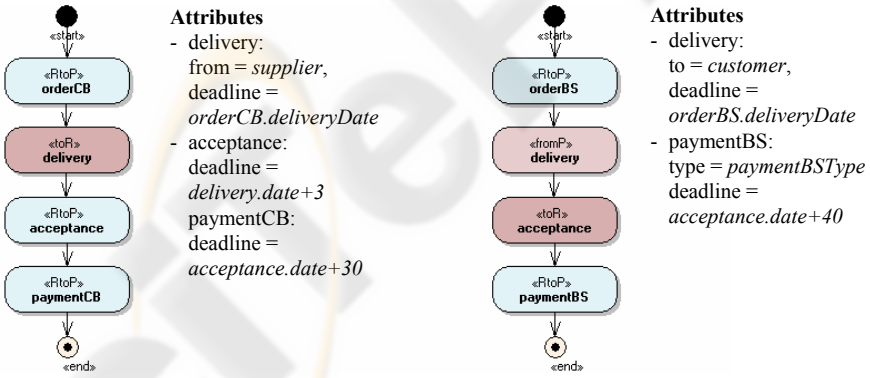


Fig. 8. The models of extended collaborations cbC (left) and bsC (right).

The extended collaboration models are shown in Fig. 8. Extended messages are represented by stereotypes “toP”, “toR”, “fromP” and “fromR”. If we combine those models by merging the interactions with the same names we come up with the global

model shown in Fig. 3, and as a consequence the business processes resulting are the same as those shown in Fig. 6.

4 Conclusion

Current work proceeds in several directions. We are making an in-depth comparison between our approach and choreography description languages such as WS-CDL [6]. WS-CDL is a complex XML-based language and lacks a graphical notation, therefore a mapping from our models to WS-CDL descriptions is under consideration; moreover, as pointed out in [7], it is not clear how WS-CDL deals with multiple collaborations. Since we follow the MDA [8] principles, a further step is to automatically produce behavioral interface models from collaboration models; the automatic mapping from business processes to BPEL processes has already been achieved [2]. Finally we want to add transactional features [9] to our collaboration models. As to modeling notations, BPMN [10] is gaining growing consensus, however it does not treat communication activities as first-level entities and, for this reason, it seems to be more adequate to workflow processes than to collaborative ones.

References

1. Andrews, T. et al.: Business Process Execution Language for Web Services (BPEL4WS), Version 1.1. BEA Systems, IBM, Microsoft, SAP AG and Siebel Systems (2003). http://www-128.ibm.com/developerworks/library/specification/ws-bpel/
2. Bruno, G., La Rosa, M.: From collaboration models to BPEL processes through service models. In Pre-proceedings of the 1st Int. Workshop on Web Service Choreography and Orchestration for Business Process Management, BPM 2005, Nancy (2005) 16-30
3. OMG: Unified Modeling Language: Superstructure, Version 2.0 (2005). http://www.omg.org/cgi-bin/doc?formal/05-07-04
4. Barros, A., Dumas, M., Oaks, P.: Standards for Web Service Choreography and Orchestration: Status and Perspectives. In Pre-proceedings of the 1st Int. Workshop on Web Service Choreography and Orchestration for Business Process Management, BPM 2005, Nancy (2005) 1-15
5. Barros, A., Dumas, M., ter Hofstede, A.H.M.: Service interaction patterns. In: van der Aalst, W.M.P., Benatallah, B., Casati, F., Curbera F. (eds.): BPM 2005. Lecture Notes in Computer Science, Vol. 3649. Springer (2005) 302-318
6. Kavantzas, N. et al. (eds): Web Services Choreography Description Language (WS-CDL), Version 1.0. W3C (2004). http://www.w3.org/TR/ws-cdl-10/
7. Barros, A., Dumas, M., Oaks, P.: A Critical Overview of the Web Services Choreography Description Language (WS-CDL). BPTrends (2005). http://www.bptrends.com
8. Mellor, S., Clark, A. N., Futagami, T.: Special Issue on Model-Driven Development. IEEE Software, Vol. 20 (5). IEEE Computer Society (2003)
9. Dalal, S., Temel, S., Little, M., Potts, M., Webber, J.: Coordinating business transactions on the web. IEEE Internet Computing, Vol. 7 (1). IEEE Computer Society (2003) 30-39
10. White, S. A.: Business Process Modeling Notation (BPMN), Version 1.0, BPMI (2004). http://www.bpmn.org