Interaction Modeling Using BPMN

Gero Decker[1] and Alistair Barros[2]

[1] Hasso-Plattner-Institute, University of Potsdam, Germany
gero.decker@hpi.uni-potsdam.de
[2] SAP Research Centre, Brisbane, Australia
alistair.barros@sap.com

Abstract. Process choreographies describe interactions between different business partners and the dependencies between these interactions. While different proposals were made for capturing choreographies at an implementation level, it remains unclear how choreographies should be described on a conceptual level. While the Business Process Modeling Notation (BPMN) is already in use for describing choreographies in terms of interconnected interface behavior models, this paper will introduce interaction modeling using BPMN. Such interaction models do not suffer from incompatibility issues and are better suited for human modelers. BPMN extensions are proposed and a mapping from interaction models to interface behavior models is presented.

1 Introduction

The Business Process Modeling Notation (BPMN [1]) is the de-facto standard for business process modeling. It is mainly used for capturing activities, decision responsibilities, control and data flow in business process *within* one organization. However, in cross-organizational settings we concentrate on the interaction behavior between the different partners involved. The individual partners can internally implement processes as they like as long as their interaction behavior conforms to the *choreography* that is agreed upon. Especially when relying on electronic messages as means for interaction between different partners, an exact definition of message formats and interaction sequences is of major importance.

BPMN can already be used for choreography modeling by expressing interconnected interface behavior models. However, this modeling style leads to redundant control flow dependencies and the danger of incompatible processes. An example for such incompatibility would be a supplier who waits for the payment to arrive before delivering the purchased goods. The buyer, on the other hand, waits for the goods to be delivered before actually paying for them. Both partners would wait endlessly – a classical deadlock situation. Interaction models avoid these problems by describing control flow dependencies between interactions. This means that a particular control flow dependency is not explicitly assigned to any of the partners in the model.

Another drawback of redundancy is that modelers need more time for creating and understanding the models. It has turned out that interaction modeling allows faster creation and understanding by human modelers.

A. ter Hofstede, B. Benatallah, and H.-Y. Paik (Eds.): BPM 2007 Workshops, LNCS 4928, pp. 208–219, 2008.

There are different language proposals for interaction modeling, e.g. the Web Service Choreography Description Language (WS-CDL [9]) and Let's Dance [11]. WS-CDL operates on an implementation level and only comes with a textual syntax. Let's Dance has a graphical notation, however, it is very different to that of established process modeling languages. The motivation for extending BPMN is to reuse a very popular notation as many process modelers are already trained in this notation.

The remainder of this paper is structured as follows. The next section will revisit choreography modeling with standard BPMN, before section 3 introduces the extensions for interaction modeling. Section 4 shows how interface behavior models can be generated out of interaction models. Section 5 will report on related work and section 6 concludes and points to future work.

2 Choreography Modeling Using Standard BPMN

A bidding scenario is going to be used as sample choreography throughout this paper. Three types of participants are involved in this scenario: a seller, several bidders and an auctioning service. The seller initiates an auction with the goal to sell her goods for the highest possible price. She does not operate the auction by herself but rather outsources this to an auctioning service. Different bidders can join in if they are interested in the goods and place their bids accordingly. Figure 1 shows the structural view on this collaboration scenario using BPMN.

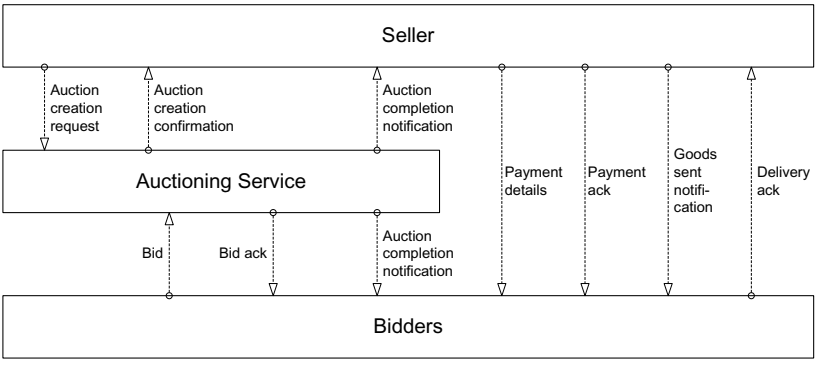

Fig. 1. Bidding scenario: Structural view

The participant types are represented by pools and message flows between the pools indicate which messages might be sent from a participant of one type to a participant of another type. Ordering constraints between the message exchanges are not expressed in the diagram.

Figure 2 depicts the complete choreography consisting of interconnected behavioral interfaces. For every message flow message send and receive events are

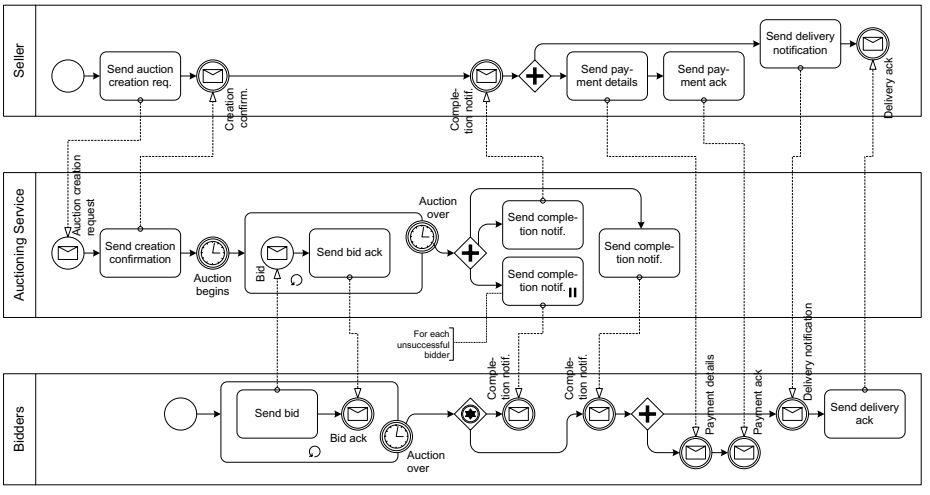

Fig. 2. Interconnected behavioral interfaces for the bidding scenario

introduced. The control flow within each pool connects these communication activities and therefore defines the behavioral dependencies between the different message exchanges.

First, the seller sends an auction creation request to the auctioning service who acknowledges it with a confirmation message. As soon as the auction begins (depicted by an intermediate timer event), bidders can place bids that are in turn confirmed by the auctioning service. The auction ends at a given point in time and the auctioning service notifies the seller about which seller has placed the highest bid and how high the corresponding amount is. The bidder who has won the auction also gets a notification. All other bidders with lower bids are informed, too. Finally, payment and shipment can happen in parallel. The seller sends payment details containing e.g. the bank account number to the successful bidder and acknowledges the payment as soon as the money has arrived. On the shipment side, the seller sends a notification to the bidder as soon as the goods are sent and the bidder acknowledges the delivery.

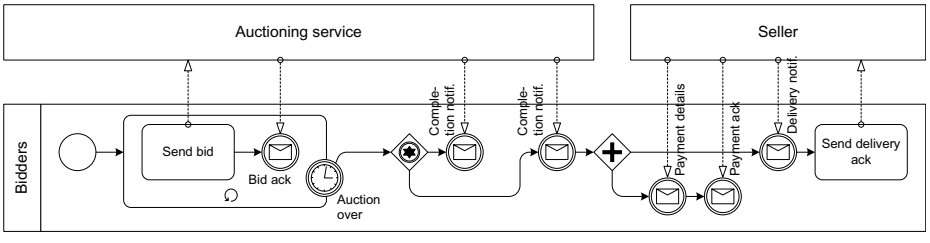

Fig. 3. Participant behavior description for bidders

While the choreography model contains all relevant interactions and dependencies, interface behavior models are the individual views on the choreography from the perspective of one of the participants. Figure 3 shows how such a model looks like for the bidders. Only the communication actions of the bidders are included, while the other participants are depicted as black boxes.

Modeling choreographies in terms of interconnected interface behavior models has two drawbacks:

1. **Redundancy.** As an example, the ordering constraint between the auction creation request and the creation confirmation appears twice: in the interface behavior models of the seller and the auctioning service. Parallelism, branching, loops and timeouts are duplicated in the model, too. This redundancy involves unnecessary modeling effort and often lead to invalid models.
2. **Potentially incompatible behavior.** If sequencing structures do not match properly, we might run into deadlocks. An even more common modeling error occurs in the case of branching: While modelers immediately understand the semantics of data-based XOR-gateways, we often find misunderstanding in the case of event-based XOR-gateways, resulting in erroneous models.

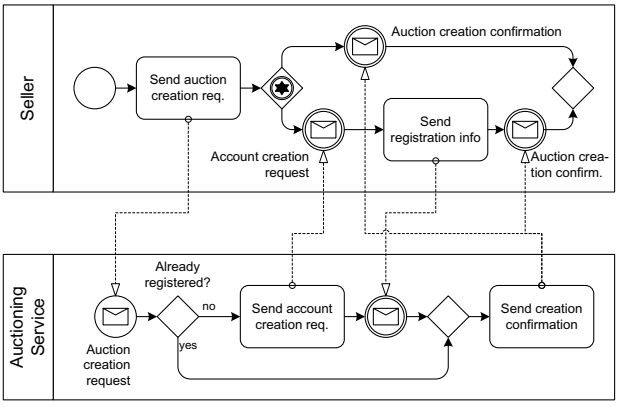

Fig. 4. Matching branching structures

Figure 4 shows properly matching branching structures. The data-based XOR-gateway on the auctioning service's side indicates that it decides whether registration is needed or not. The event-based XOR-gateway makes the choice on the seller's side dependent on which message comes in. Process modelers often use data-based gateways instead of event-based gateways, ignoring the location of where the choice is made. Furthermore, we see in this example that the receipt action for the auction creation confirmation needs to appear twice in the seller's interface behavior model. Such problems compound when further parties are involved. Also, looping and multiple instances are typical sources for mismatches between the different interface behavior models.

Interaction models do not suffer these drawbacks. As interactions are the basic building blocks, less nodes are needed to express the same choreography as an interaction model. Especially the distinction between local choices and choices made by the environment is often not needed. Incompatibility does not occur as control flow dependencies are not duplicated. While observing human modelers using the different choreography modeling styles, it turned out that interaction modeling leads to faster model creation and understanding.

However, interaction models come with their own anomalies. Locally unenforceable choreographies, i.e. choreographies where the individual participant cannot collectively enforce global control flow constraints without additional synchronization messages, can be expressed. Imagine e.g. that an interaction between the seller and a bidder must only occur after a certain interaction between the auctioning service and another bidder has taken place. In this case the seller and the first bidder cannot know when the second interaction has actually happened. This property of local enforceability is reported in [12] and [5].

3 BPMN Extensions for Interaction Modeling

This section is going to introduce the BPMN extensions for interaction modeling, which we will call "iBPMN". Atomic interactions are going to be the basic building blocks of these models and control and data flow are defined between them. I.e., we do not use separate send and receive activities in the models.

Each elementary interaction (represented as message event) is attached to a message flow in iBPMN, as shown in Figure 5. All control flow constructs that are available in BPMN also apply in these interaction models. E.g. we see that parallelism and timers appear in the choreography. Elementary interactions

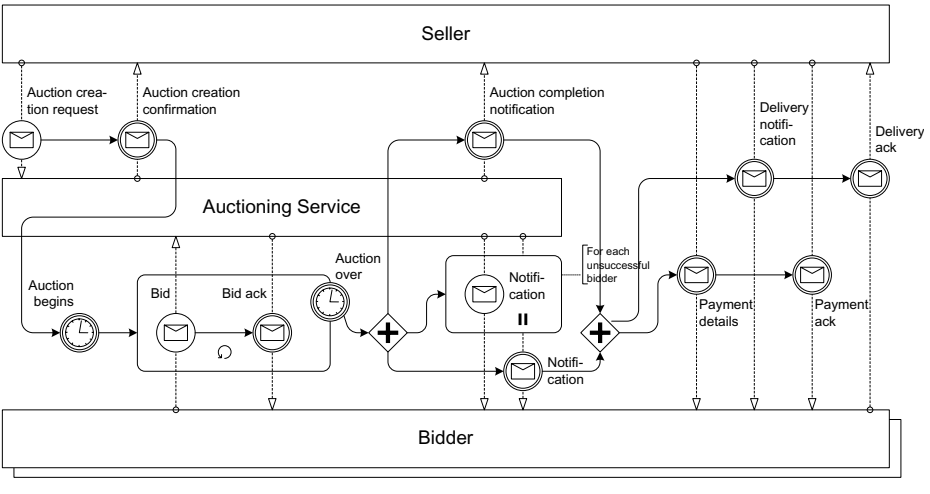

Fig. 5. iBPMN interaction model for the bidding scenario

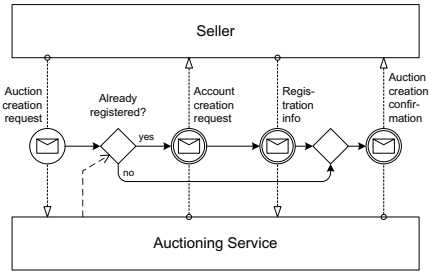

Fig. 6. Explicit choice

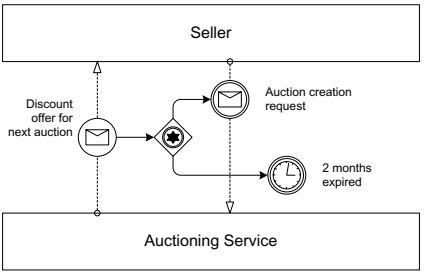

Fig. 7. Racing choice

can be composed to form complex interactions, enabling loop interactions and multiple instances interactions as shown in the figure.

In iBPMN, pools are empty, i.e., the internal behavior of the participants is completely hidden. Therefore, the interaction model is a refinement of the structural diagram in Figure 1.

A recurrent scenario in choreographies is that several participants of the same type are involved in one conversation. Our bidding scenario also includes this case: Several bidders participate in an auction. In order to make a clear distinction between the case with only one participant of a type vs. potentially many participants, we introduce shadowed pools as shown in Figure 5.

In interaction models we make a distinction between explicit choices and racing choices. In the case of explicit choices one participant decides which branch to take. This is represented by a data-driven XOR-gateway. We further add an association between the gateway and one of the pools in order to define who actually carries out the choice. In the case of racing choices one among a set of events can happen and the event occurring first inhibits the others from happening and determines which branch is taken. Event-based XOR-gateways depict this. Figures 6 and 7 illustrate the two types of choices.

Another recurrent scenario in choreographies is passing on participant references. Imagine the payment between bidder and seller is carried out using a payment service and the seller can choose which service to use. The seller needs to pass on the reference to this service to the bidder so that the bidder can issue the payment with that service. Figure 8 illustrates how this is represented in iBPMN: A data object is attached to the message flow and the object is in turn associated with the corresponding participant.

We are now going to validate the suitability of iBPMN for choreography modeling by investigating which of the Service Interaction Patterns are directly supported. These patterns describe recurrent scenarios in choreographies and have already been used to assess Let's Dance [11] and WS-CDL [4].

Some of these patterns appear in our sample choreography. E.g., *Send*, *Receive* and *Send/receive* can be found in the first two interactions where the seller initiates the auction.

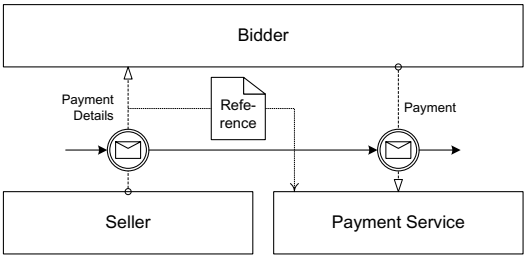

Fig. 8. Participant reference passing

In the case of *Racing incoming messages* a participant processes the first out of a set of messages that he receives. This can be modeled using an event-based XOR-gateway. *One-to-many send* occurs in the choreography where the auctioning service sends out notifications to all the unsuccessful bidders, which is modeled through a multiple instances send activity. However, the fact that a message is sent to all unsuccessful bidders is only captured by the annotation. It might be desirable to more directly integrate such a "for each" into iBPMN. A general drawback of BPMN is that it cannot be specified which particular participant a message is sent to, only the participant type is defined. We have said that simple pools indicate that there must be at most one participant of that type in one choreography instance. Therefore, we can be sure that all messages sent to a participant of type auctioning service are actually sent to the same concrete participant, if all messages belong to the same choreography instance. It becomes difficult in those cases where we have many participants of the same type in one choreography instance. In our example we do not directly see that there is a distinction between the bidder with the highest bid and the remaining bidders.

One-from-many receive can be found during the bidding phase: The auctioning service does not know in advance how many bidders are going to take part in the auction. As there are potentially many bidders a bid from any sender is received and processed. *Multi-responses* is a bi-lateral pattern where several responses are sent back as result of a single request. This can easily be modeled using loop interactions in iBPMN. *Contingent requests* involves a list of recipients for requests. If the first recipient does not respond within a given timeframe, the request is sent to the second and so on. Loop interactions with corresponding annotations express this in iBPMN. However, late responses from previous participants are discarded. Therefore, this pattern is only partially supported in iBPMN. *Atomic multicast notification* is not supported in iBPMN.

Relayed request requires that a participant observes a conversation between two other participants. This can easily be modeled using two parallel interactions in iBPMN. *Request with referral* alludes the notion of link passing mobility. Figure 8 showed how this is modeled in iBPMN.

We see that all Service Interaction Patterns are directly supported in iBPMN (except *atomic multicast notification*). In this sense iBPMN provides the same pattern support as Let's Dance. iBPMN provides better pattern support than

WS-CDL, as scenarios where multiple participants of the same type are involved and the exact number of participants are only known at design-time are fully supported in iBPMN.

4 Generation of Interface Behavior Models

While deriving individual interface behavior models from classical BPMN choreographies is trivial (see Section 2), deriving these models from iBPMN choreographies is more complex. A typical approach to generating interface behavior models out of interaction models is by means of model reduction (cf. [12]). Those interactions where the corresponding participant is not involved in are marked as τ-actions and they are removed from the model while preserving control flow dependencies. This section presents an algorithm for interface behavior models out of simple iBPMN models.

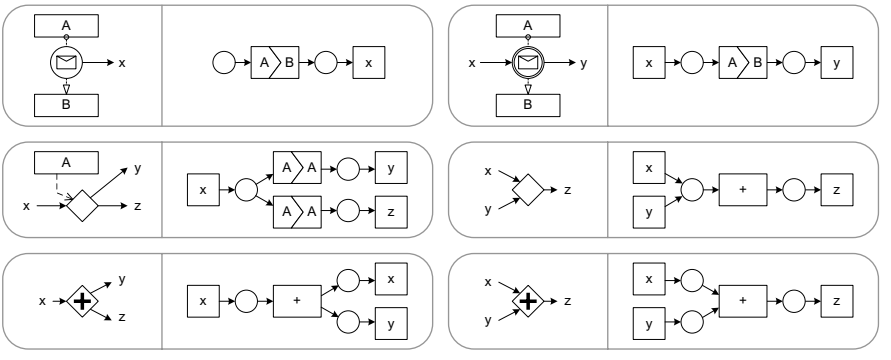

Fig. 9. iBPMN constructs and their interaction Petri net representation

We restrict the algorithm to a very small subset of iBPMN models. We only allow elementary interactions as well as AND- and data-based XOR-gateways. In [5] we have already presented a reduction algorithm for interaction Petri nets, an extension to classical place / transition nets for interaction modeling. We are going to reuse this algorithm in the following way:

1. Translate the simple iBPMN model to an interaction Petri net. Figure 9 shows the translation rules for the allowed constructs. We mark those interactions as τ-actions where the participant who we generate the interface behavior model for does not participate. We also mark those transitions representing exclusive choices being made by another participant as τ. The transitions representing AND-gateways are labeled "+".
2. Apply the reduction algorithm from [5]. This removes all τ-transitions from the model. Those "+"-transitions that are involved in choices, i.e. sharing a common input place with another transition, are relabeled to τ and are reduced as well.

3. Transform the resulting interaction Petri net in such a way that only those patterns appear that can be directly translated back to BPMN. Optionally, the net can be reduced by removing redundant places or removing "+"-transitions with at most one preceding and at most one succeeding transition.
4. Translate the interaction Petri net to BPMN.

The resulting BPMN interface behavior models will contain message send activities and event-based gateways, in addition to those constructs allowed in the input iBPMN model. The interaction Petri net patterns that are translated back to BPMN are shown in Figure 10.

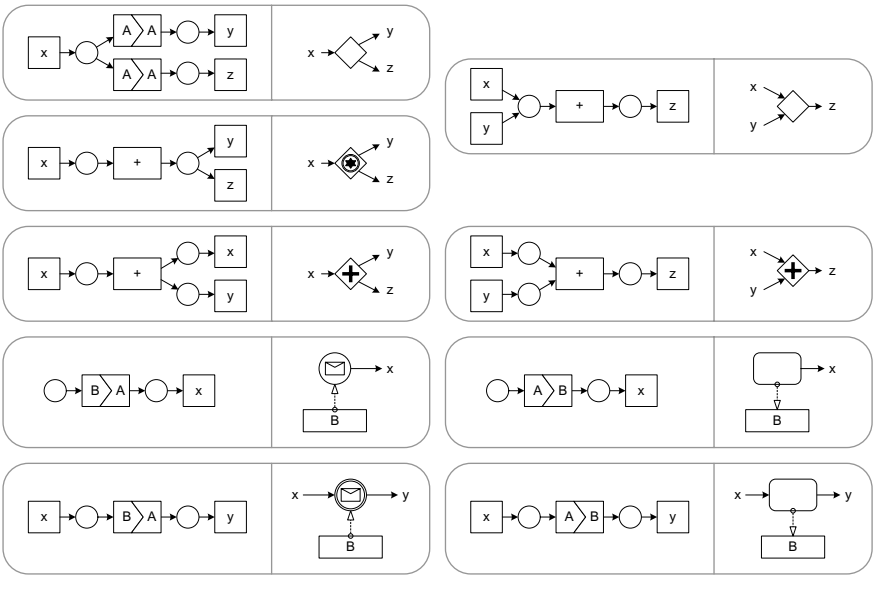

Fig. 10. Interaction Petri net patterns translated to BPMN

The transformation in step 3 leads to adding "+"-transitions in those cases where interactions have more than one input or output place or where input or output place are shared with other transitions. The "+"-transitions will be translated to gateways later on. Only the patterns depicted in Figure 10 can be translated. Therefore, the introduction several "+"-transitions might be necessary in some structures.

Figure 11 shows the result of the generation algorithm. The two interactions $m3$ and $m5$ do not occur in the interface behavior model for A, as this participant is not involved in $m3$ and $m5$. The choice made by B results in the occurrence of an event-based gateway for A. A only knows which branch B has chosen as soon as it gets one of the messages $m2$, $m4$ and $m6$. Furthermore, $m4$ and $m6$ needed to be sequentialized as BPMN requires that an event-based gateway is followed by events as opposed to other gateways. In this case, the parallelism from the

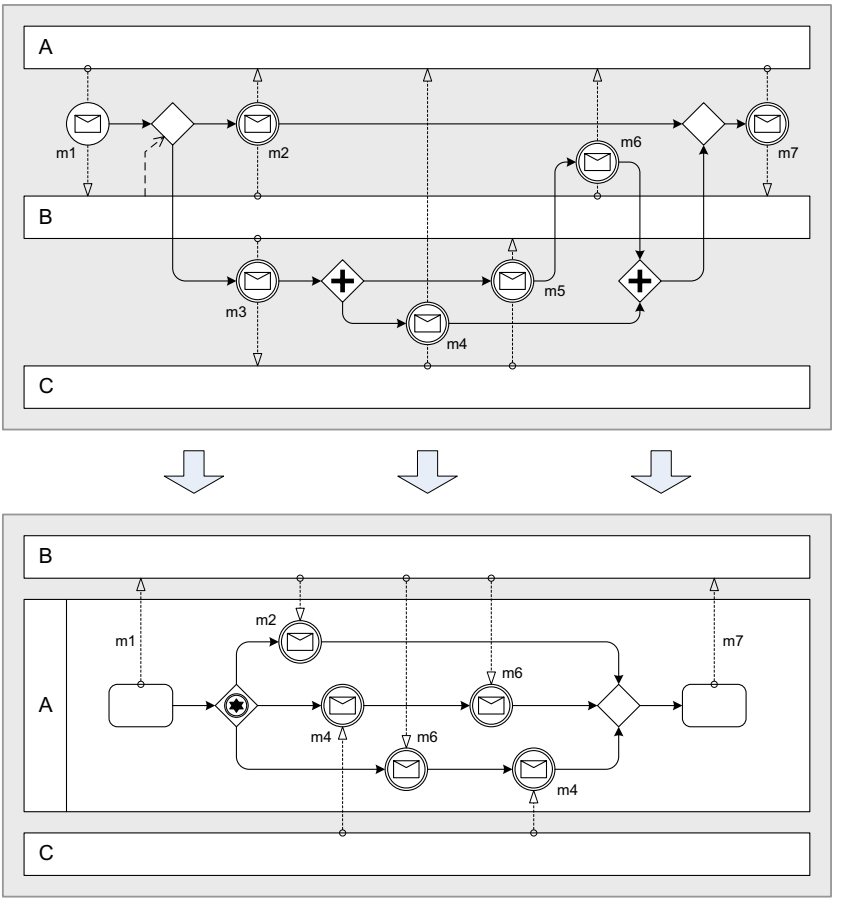

Fig. 11. Generation of a BPMN interface behavior model from an iBPMN model

interaction model completely disappears in A's interface behavior model (while it would be preserved in the interface behavior model for C).

It is not possible to generate valid BPMN interface behavior models out of every valid simple iBPMN model. This is due to the fact that the reduction algorithm might produce non-free-choice interaction Petri nets. In free-choice nets, every transition that shares an input place with another transition t, has the same set of input places like t (cf. [6]). A non-free-choice net would need to be represented by a BPMN model, where an event-based XOR-gateway is followed by an AND-gateway. This is not allowed.

5 Related Work

In [11], Zaha et al. identify the need for describing choreographies on a conceptual level and introduce the choreography language Let's Dance. It provides

direct support for most of the Service Interaction Patterns [2], a catalog of common scenarios in choreographies and therefore a benchmark for assessing choreography languages. Let's Dance follows the interaction modeling approach, i.e. interactions are the basic building block in choreography models. Let's Dance comes with a set of own control flow constructs different to those e.g. known from BPMN or UML 2.0 Activity Diagrams. BPMN [1] has been assessed for its suitability for process modeling in [10]. However, choreography modeling was not discussed and the Service Interaction Patterns were not considered.

Message Sequence Charts (MSCs [8]) can also be used for describing choreographies following the interconnected interface behavior modeling approach. However, they are rather suited for describing mere sequences of interactions in contrast to full choreographies: conditional branching, parallel branching and iterations are not supported.

WS-CDL [9] and BPEL4chor [3] are proposals for describing choreographies at an implementation level. Both approaches allow to specify choreographies of web services and do not come with a graphical representation. While WS-CDL follows the interaction modeling approach, BPEL4chor allows to specify interconnected behavioral interfaces. BPEL4chor distinguishes between three different artifact types: Participant topology, behavioral interfaces and participant grounding. The topology describes the structural aspects of the choreography, the behavioral interfaces describe the control and data flow dependencies between the communication activities within the participants and the participant grounding introduces web-service-specific configurations, e.g. the mapping of message links to WSDL port types and operations.

Dijkman et al. have defined a mapping from BPMN to Petri nets in [7]. They consider more constructs than we have used in section 4 including subprocesses, timer events and intermediate events attached to activities (cancellation).

6 Conclusion

This paper has introduced iBPMN, a set of extensions to the Business Process Modeling Notation for interaction modeling. Following an interaction modeling approach as opposed to modeling interconnected interface behavior models, it can be expected that choreography designers can understand models better, introduce less errors, such as incompatibility, into the models and are faster at creating the models. However, a detailed survey validating these hypotheses is left to future work.

We have shown that most Service Interaction Patterns can be expressed using iBPMN and we have presented an algorithm for deriving interface behavior models from simple interaction models. The algorithm is validated through ongoing implementation.

It turns out that some choreographies are not *locally enforceable*, i.e., it is possible to introduce control flow dependencies between interactions that cannot be collectively enforced by the participants without the addition of synchronization interactions. This property was reported in [12]. Verifying the absence of such

anomalies in choreographies is beyond the scope of this paper and are also left to future work.

References

1. Business Process Modeling Notation (BPMN) Specification, Final Adopted Specification. Technical report, Object Management Group (OMG) (February 2006)
2. Barros, A., Dumas, M., ter Hofstede, A.: Service Interaction Patterns. In: van der Aalst, W.M.P., Benatallah, B., Casati, F., Curbera, F. (eds.) BPM 2005. LNCS, vol. 3649, pp. 302–318. Springer, Heidelberg (2005)
3. Decker, G., Kopp, O., Leymann, F., Weske, M.: BPEL4chor: Extending BPEL for Modeling Choreographies. In: Proceedings International Conference on Web Services (ICWS) (2007)
4. Decker, G., Overdick, H., Zaha, J.M.: On the Suitability of WS-CDL for Choreography Modeling. In: EMISA 2006. Proceedings of Methoden, Konzepte und Technologien für die Entwicklung von dienstebasierten Informationssystemen, Hamburg, Germany (October 2006)
5. Decker, G., Weske, M.: Local Enforceability in Interaction Petri Nets. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, Springer, Heidelberg (2007)
6. Desel, J., Esparza, J.: Free Choice Petri Nets. Cambridge Tracts in Theoretical Computer Science, vol. 40. Cambridge University Press, Cambridge (1995)
7. Dijkman, R.M., Dumas, M., Ouyang, C.: Formal semantics and automated analysis of BPMN process models. In: Preprint 7115. Queensland University of Technology, Brisbane, Australia (2007)
8. ITU-T. Message sequence chart. Recommendation Z.120, ITU-T (2000)
9. Kavantzas, N., Burdett, D., Ritzinger, G., Lafon, Y.: Web Services Choreography Description Language Version 1.0, W3C Candidate Recommendation. Technical report (November 2005), `http://www.w3.org/TR/ws-cdl-10`
10. Wohed, P., van der Aalst, W.M., Dumas, M., ter Hofstede, A., Russell, N.: On the Suitability of BPMN for Business Process Modelling. In: Dustdar, S., Fiadeiro, J.L., Sheth, A.P. (eds.) BPM 2006. LNCS, vol. 4102, Springer, Heidelberg (2006)
11. Zaha, J.M., Barros, A., Dumas, M., ter Hofstede, A.: A Language for Service Behavior Modeling. In: CoopIS 2006. Proceedings 14th International Conference on Cooperative Information Systems, Montpellier, France (November 2006)
12. Zaha, J.M., Dumas, M., ter Hofstede, A., Barros, A., Decker, G.: Service Interaction Modeling: Bridging Global and Local Views. In: EDOC 2006. Proceedings 10th IEEE International EDOC Conference, Hong Kong (October 2006)