Do We Need Internal Behavior in Choreography Models?

Oliver Kopp and Frank Leymann

Institute of Architecture of Application Systems, University of Stuttgart, Germany
Universitätsstraße 38, 70569 Stuttgart, Germany
{lastname}@iaas.uni-stuttgart.de

Abstract. Choreographies capture the message exchanges between multiple processes. Certain choreography languages ignore the internal behavior completely, other languages offer the possibility to model internal behavior. This paper presents an example modeled in both types of languages and discusses the need to integrate internal behavior in choreographies.

1 Introduction

A choreography defines the message exchanges between several processes. While it is fundamental that a choreography language has to include the ability to express message exchanges, it is an open question whether a choreography language should offer constructs to model internal behavior. In this paper, we present a sample choreography and use this example to discuss, whether internal behavior should be modeled in a choreography.

The example is introduced in Sect. 2. Afterwards, it is modeled in Let's Dance (Sect. 3) and BPMN (Sect. 4). An overview on the support of modeling internal behavior of the most prominent choreography modeling languages is given in Sect. 5. Finally, Sect. 6 provides a conclusion and points out future work.

2 Drop-dead Order Scenario

Fig. 1 presents the drop-dead order scenario, where a customer requests a distributor to ship products until a given drop-dead date. The distributor neither produces the products by himself nor owns a delivery. Therefore, the distributor asks a supplier whether he can produce the requested products in time and the distributor asks a carrier whether he can carry the produced products by time. If the supplier and the carrier agree, then the customer's order is accepted and rejected otherwise. The example itself was first presented in [1] and is used in [2] to develop transaction requirements for services. We use this example in this paper to illustrate the difference between different choreography languages.

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073

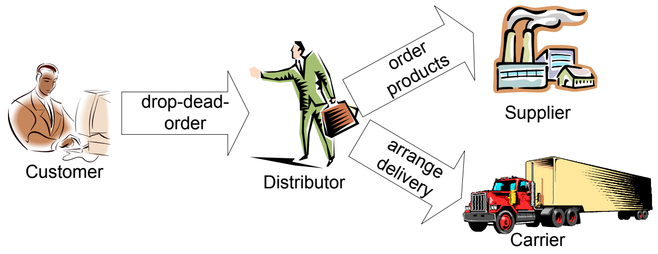

Fig. 1. Drop-dead oder (adapted from [1])

3 Let's Dance Model

Let's Dance [3] is a choreography language having the interaction as basic building block. Therefore, the model is called "interaction model" [4]. A choreography model of the drop-dead order scenario is presented in Fig. 2. The topmost box denotes that a customer sends an order to a distributor. Control flow is modeled using directed arcs. The first arc points to a group, labeled with a circled A in the figure. In this group, the message exchange between the customer and the shipper and between the customer and the carrier happens in parallel. The crossed line between the acceptance and the rejection denotes that exactly one of the two message exchanges may happen. After both the supplier and the carrier have sent an answer, they have either accepted or rejected. Afterwards, group B is active. In case both the supplier and the carrier accepted the request of the distributor, they receive an acceptance of the distributor and the products are built and delivered to the customer. Otherwise, the order is rejected at the supplier and the carrier. The client's order is rejected, too.

In the presented choreography, there is no internal behavior modeled. The messages in the choreography suggest that the supplier is somehow producing products and that the carrier delivers them. However, there is no explicit description of these tasks.

4 BPMN Model

Figure 3 presents the drop-dead order scenario using BPMN [5]. BPMN is a choreography language belonging to the category of interconnection models [4]. In interconnection models, the control flow is modeled per participant. While the interaction is one building block in interaction models, the interaction is split up in send and receive activities in the case of interconnection models. To provide a better overview on the model, one path through the model is highlighted. This path denotes the case, where both the supplier and the carrier accept the request of the distributor. First, the customer sends an order to the distributor, which asks the supplier and the carrier in parallel, whether they can produce/deliver in time. The supplier and the carrier decide and both accept the order. Since both have accepted, they are notified that the distributer accepted, too. Finally, the

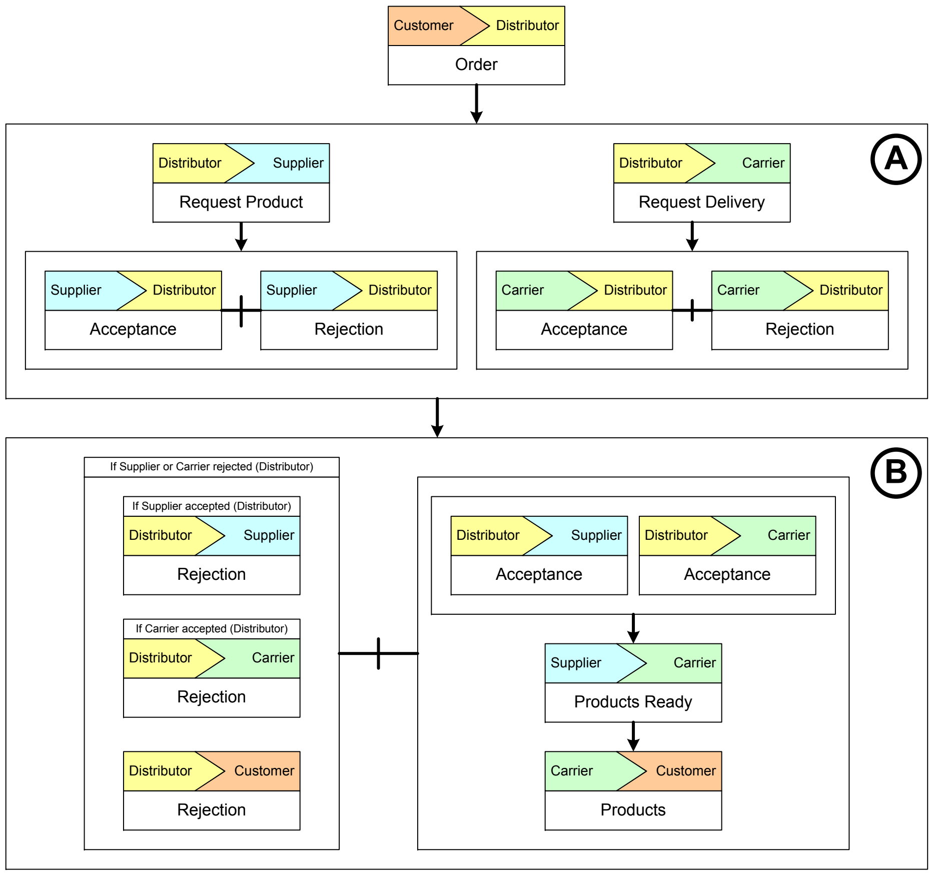

Fig. 2. Drop-dead order scenario modeled in Let's Dance

producer produces, notifies the carrier, which picks up the products and delivers them to the customer.

Besides the split of the interactions into interconnections of send and receive activities, two internal activities of the supplier and the carrier are shown. In case of the supplier, the choreography model shows the decision activities and the produce activity. When it comes to implement a supplier based on the choreography description, the internal behavior can be used as basis for an executable BPEL process [6]. In case the choreography model is detailed enough, an IT export just has to add concrete WSDL information to the activity to turn the process into an executable BPEL process.

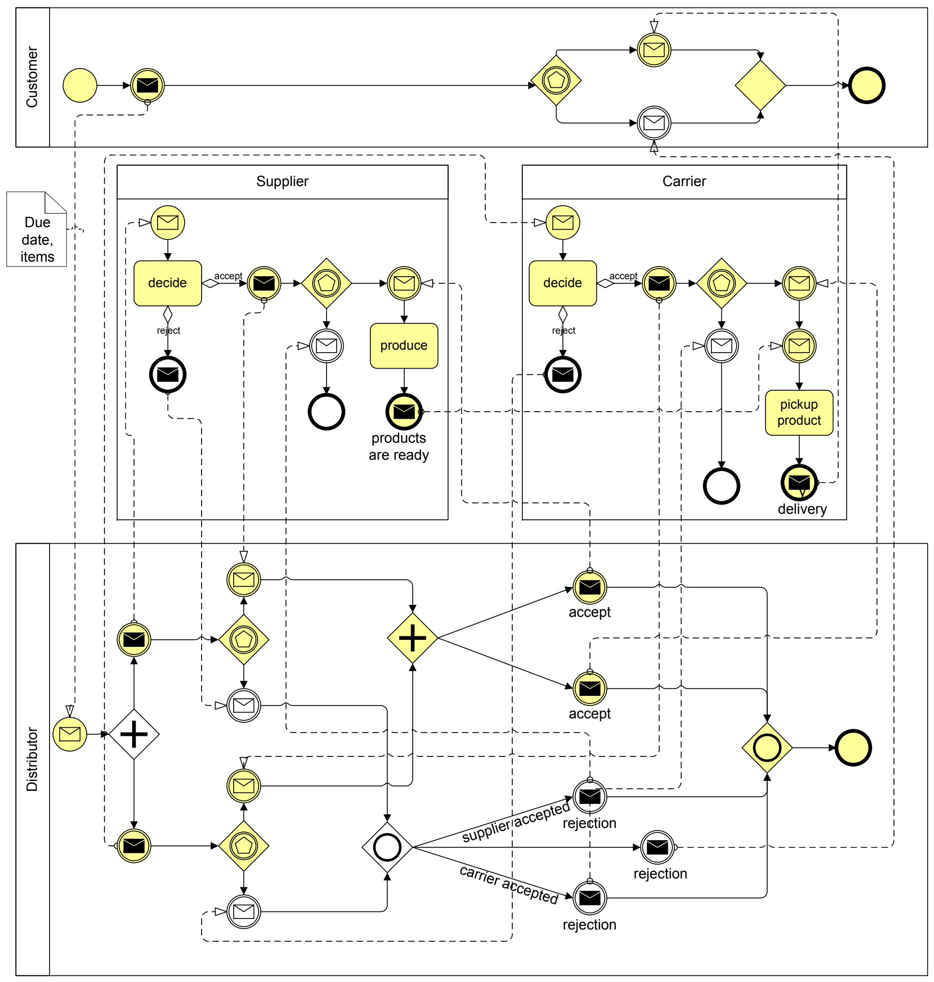

Fig. 3. Drop-dead order scenario modeled in BPMN

5 Support of Internal Actions

Language	Model Type	Constructs for Internal Behavior Available?
BPEL4Chor	interconnection	+
BPMN 1.2	interconnection	+
iBPMN	interaction	–
Let's Dance	interaction	–
WS-CDL	interaction	+

Table 1. Support of modeling internal actions in choreography languages

Table 1 lists five choreography languages and their support of internal actions. The languages are chosen, because they are the most prominent choreography languages. The second column denotes whether the language allows to model interaction models or interconnection models.

BPEL4Chor [7] is a choreography language extending BPEL with choreography constructs. Since BPEL is re-used, so called opaque activities can be used to model internal behavior. The Business Process Modeling Notation Version 1.2 (BPMN 1.2) has been used in Sect. 4 to model the drop-dead order example. It has been shown that this language supports internal behavior. iBPMN [8] is an extension for BPMN to support interaction modeling. Each message exchange is atomic and does not have to be modeled with two distinct send and receive operations. iBPMN does not support the inclusion of internal behavior. It is shown in Sect. 3 that Let's Dance does not support the inclusion of internal behavior. The Web Services Choreography Language (WS-CDL, [9]) provides the "silent action activity" to express internal behavior. Thus, WS-CDL is the only language following the interaction approach and providing support for internal actions.

6 Conclusion and Outlook

In this paper, we gave an overview on the support of modeling internal behavior in choreography languages. The drop-dead order scenario was introduced and modeled using Let's Dance and BPMN. Finally, we showed the support of internal actions in five choreography languages. All languages supporting interconnection models also support the modeling of internal behavior. In the case of interaction models, two of the three languages do not support modeling internal behavior. WS-CDL is the only language based on interaction models, where internal behavior can be modeled.

The examples suggest that it depends on the use-case of the choreography whether internal actions are needed. If the choreography is used to serve a basis for executable processes, the internal actions can be seen as placeholders for calls to internal services. Thus, the effort to build executable BPEL processes seems less. We are going to describe concrete scenarios and to use them as basis for a comparison.

Acknowledgments This work is supported by the BMBF funded project Tools4BPEL (01ISE08B).

References

1. Haugen, B., Fletcher, T.: Multi-Party Electronic Business Transactions. Technical report, UNCEFACT (2002)
2. Sun, C., Aiello, M.: Requirements and Evaluation of Protocols and Tools for Transaction Management in Service Centric Systems. In: 31st Annual International Computer Software and Applications Conference (COMPSAC 2007), IEEE Computer Society (2007) 461–466
3. Zaha, J.M., Barros, A.P., Dumas, M., ter Hofstede, A.H.M.: Let's Dance: A Language for Service Behavior Modeling. In: CoopIS 2006: Proceedings 14th International Conference on Cooperative Information Systems. Volume 4275 of Lecture Notes in Computer Science., Springer (2006) 145–162
4. Decker, G., Kopp, O., Barros, A.: An Introduction to Service Choreographies. Information Technology **50**(2) (2008) 122–127
5. Object Management Group (OMG): Business Process Modeling Notation (BPMN) Version 1.2. (2009) `http://www.bpmn.org/`.
6. OASIS: Web Services Business Process Execution Language Version 2.0 – OASIS Standard. (2007)
7. Decker, G., Kopp, O., Leymann, F., Weske, M.: BPEL4Chor: Extending BPEL for Modeling Choreographies. In Society, I.C., ed.: Proceedings of the IEEE 2007 International Conference on Web Services, IEEE Computer Society (2007) 296–303
8. Decker, G., Barros, A.P.: Interaction Modeling Using BPMN. In ter Hofstede, A.H.M., Benatallah, B., Paik, H.Y., eds.: CBP: Proceedings of the 1st International Workshop on Collaborative Business Processes. Volume 4928 of Lecture Notes in Computer Science., Springer (2007) 208–219
9. Kavantzas, N., Burdett, D., Ritzinger, G., Lafon, Y.: Web Services Choreography Description Language Version 1.0. W3C. (2005) `http://www.w3.org/TR/ws-cdl-10`.

All links were followed on 2009-02-19.