# REST-Enabled Decision Making in Business Process Choreographies

Adriatik Nikaj(✉), Kimon Batoulis, and Mathias Weske

Hasso Plattner Institute at the University of Potsdam, Potsdam, Germany
{adriatik.nikaj,kimon.batoulis,mathias.weske}@hpi.de

**Abstract.** In the field of business process management, the interaction between business actors or services are modeled via business process choreographies. However, enforcing or implementing business process choreographies is a challenge particularly related to the choreography's exclusive gateways, which are used to model shared decisions among business actors. Since there is no central locus of control, participants may interpret the data relevant for decision making differently. To tackle this problem, this paper offers a solution by delegating the decision making to a decision service. This service is based on the recently published Decision Model and Notation standard and is provided to the choreography participants via a REST interface. The RESTful decision service assures a correct implementation of choreographies' exclusive gateways and provides a blueprint for RESTful services that offer decision-making solutions based on the DMN standard.

**Keywords:** Process choreographies · DMN · RESTful interactions

## 1 Introduction and Motivation

Business process choreography [1] is an intrinsic part of business process management (BPM) [2]. It is a modeling language for specifying interactions between business actors from a global perspective. It borrows, for a good part, a set of modeling components from business processes. While reusing modeling elements from business processes for choreographies is a good design approach, it does not come without drawbacks. Such a modeling element is the exclusive gateway.

Exclusive gateways are a key modeling construct in business processes. They are used to model alternative paths in the control flow based on the value of some data, e.g., if payment is completed send the shipment, otherwise send a reminder. They are also reused in choreographies to model alternative paths in the interaction between participants, e.g., the choice of the payment methods for a given service leads to different sequence of interactions between the client and the service. Nevertheless, in business process choreographies, differently from business processes, there is no central locus of control that is able to store and maintain the data used for the decision across participants.

Therefore, different constraints are introduced in BPMN 2.0 [1] for using exclusive gateways in choreographies. The data, once shared, does not change


Q.Z. Sheng et al. (Eds.): ICSOC 2016, LNCS 9936, pp. 547–554, 2016.
DOI: 10.1007/978-3-319-46295-0_34

until the point where the decision is taken. Thus, each participant takes its decision on the same data. However, it is assumed that the participants affected by the exclusive gateway have the same understanding of the data. We argue this strong assumption is a problem when it comes to the interaction of participants that represent companies or services which are different, e.g., in terms of area of expertise, domain, background and country.

In this paper we propose a solution to this problem by delegating the decision making to a decision service that uses the Decision Model and Notation (DMN) standard [3]. Using a standard, assures that the decision is taken properly by all the involved participants. Additionally, we introduce a generic RESTful API [4] for such a decision service that is responsible for executing the shared decision on behalf of the choreography's participants. After describing the decision service's generic RESTful API, we provide a procedure on how to embed the decision service into a RESTful choreography [5]—an extension of business process choreography with REST notation.

This way, we make sure that the choice made by each participant leads only to intended paths and introduces no deadlocks. Additionally, a higher degree of separation of concerns for decision and process logic is achieved, improving aspects such as the choreography's comprehensibility and maintainability [6].

This paper is structured as follows. Section 2 states the problem using a running example. Section 3 describes the proposed solution while Sect. 4 concludes the paper.

## 2 Problem Statement

Business process choreographies are introduced in the BPMN 2.0 specification [1]. They serve the purpose of modeling interactions between two or more business actors from a global perspective. More specifically, they abstract from internal business process activities and focus only on the messages exchanged between participants with the aim of reaching a common goal.

Figure 1 depicts an example of a choreography diagram that describes the interaction of a manufacturer with its suppliers. The manufacturer makes a request for tender to different suppliers for the product part it needs. The suppliers follow up by sending their offers. After receiving all offers, the manufacturer announces the score which represent the level of satisfaction for each supplier's offer. If there is at least a single score which passes a threshold then the supplier with the best score receives the payment from the manufacturer and sends the product part. On contrary, the tender is not successful and the suppliers are asked to send again their offers. This tender can be closed at anytime during this loop but it is not shown explicitly in the choreography model for simplification purposes.

In choreography diagrams exclusive gateways model alternative paths. However, choreographies' exclusive gateways are constrained in their usage compared to their respective counterparts in business processes. In order for the choreography to be enforceable the following constraints should hold: The data

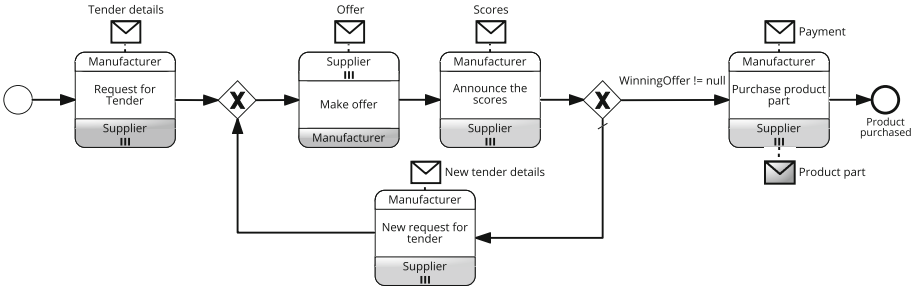


**Fig. 1.** A business process choreography for parts procurement by a manufacturer

used for the gateway conditions must have been in a message sent at some point in the choreography before the gateway. The message(s) containing the data is sent or received by all participants that are affected by the gateway and any change of the data must be visible to all these participants. And lastly, every participant must interpret the data in the same way.

However, there is a problem in implementing such constraints when it comes to the interaction of participants whose role can be filled by many possible business actors. The problem consists in that these different business actors can have different understandings of the data used for the decision making, leading the choreography to be out of sync. This is due to the fact that the business actors might be very diverse in terms of, e.g., domain and country. To ensure the enforceability of the choreography, we need to a go a level closer towards the implementation level. To this end, we use RESTful choreography—an extension of business process choreography with REST implementation information [5].

In order to make the decision clear for all participants we use DMN. Figure 2 shows an example decision model; *decisions* are rectangles, *input data* are ellipsis, *information requirement edges* are solid, and *knowledge sources* are rectangles with a wavy bottom. The decision element is associated with a FEEL (Friendly Enough Expression Language) expression displayed as an annotation next to it.

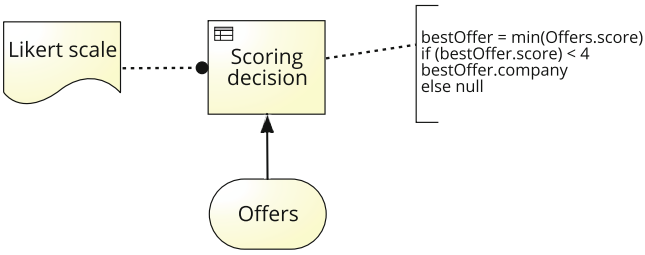


**Fig. 2.** Decision model used by the manufacturer to decide on a supplier (cf. right-most gateway in Fig. 1)

This example is based on the choreography in Fig. 1. More particularly, this is the decision model employed by the manufacturer to decide which supplier to choose from all the suppliers that made an offer. The decision element is labeled *Scoring decision* and takes as input data a list of *Offers*. These *Offers* have two attributes: *company* and *score*, where *company* is the name of the company which made the offer and *score* is the score that was assigned to that offer.

As one can see from Fig. 1, there are two possible outcomes: either there is no winning offer such that a new tender is requested, or one of the offers was chosen and the product parts are purchased from the respective supplier. The fact whether or not there is a winning offer depends on the scores, which can range from 1 to 5, according to the well known Likert scale. This scale says that the lower the score, the better. Therefore, the FEEL expression associated with the decision element first determines the offer with the minimum score. Then it checks if the score is less than four. If that is the case the respective company's name is returned. Otherwise, the decision yields a *null* value.

## 3 RESTful Decision Service

In this chapter we introduce the concept of RESTful decision service and how this service is used in conjunction with RESTful choreographies.

### 3.1 REST Interface Based on DMN

To solve the problem of the data misinterpretation we propose a RESTful decision service that is built on the DMN standard. The RESTful decision service provides a REST interface for creating, managing, and executing shared decisions. The decision owner is responsible for creating the decision logic and providing it to the decision service. Thus, we have a single decision making for all the participants to execute. The decision service is responsible for computing the decision each time it is called from the participants who are affected by the exclusive gateway. The output of the decision should comply to the conditions of the sequence flows originating from the exclusive gateway, hence, making it trivial for the other participants to relate the decision output with the correct path in the choreography.

For creating the RESTful API of the decision service, we map a REST interface to the main concepts of DMN. As explained in Sect. 2, these main concepts are the decision logic, inputs, outputs and execution. The REST interface is designed according to RESTful design rules [7] and design patterns [8]. A RESTful decision service provides the REST interface described in Table 1.

The creation of a new decision is the only action that is not included in the choreography diagram because it is performed just once by the decision owner. The decision owner is the participant of the choreography that is responsible for creating the decision logic like the manufacturer in Fig. 1. The choreography diagram models the interaction that the participants have with a particular decision logic, e.g., *Scoring decision* from Fig. 2. Hence, performing this action is a requirement for executing the choreography.

**Table 1.** The interface of a RESTful decision service

| Decision action | REST request and response |
|---|---|
| Create a new decision (logic) | PUT /decisionName |
| | ⇐ HTTP/1.1 200 OK |
| | {location: /decisionName} |
| Create a decision instance | ⇒ POST /decisionName/ |
| | ⇐ HTTP/1.1 201 Created |
| | {Location: /decisionName/id |
| | Link: /decisionName/id/inputs |
| | Link: /decisionName/id/execute} |
| Insert the decision inputs | ⇒ PUT /decisionName/id/inputs/inputName |
| | ⇐ HTTP/1.1 200 OK |
| | {location: /decisionName/id/inputs/inputName} |
| Read the decision inputs | ⇒ GET /decisionName/id/inputs/inputName |
| | ⇐ HTTP/1.1 200 OK |
| | {location: /decisionName/id/inputs/inputName} |
| Execute the decision | PUT /decisionName/id/execute |
| | ⇐ HTTP/1.1 200 OK |
| | {Link: /decisionName/id/output} |

Anytime a new instance of the choreography gateway has to be executed, a new decision URI that corresponds to that gateway instance should be created by the decision owner, e.g., a new scoring decision id should be created for any tender that is requested and eventually decided at the gateway. It is important to distinguish between different decision instances because the participants should have access only to the decision instance that affects their behavior. For example, when *winningOffer* = *null* a new tender is created and, therefore, a new decision instance has to be created. In this case the participants will distinguish the new tender from the old one from the new decision URL. The decision owner inserts the decision inputs after the decision instance is created. Providing the decision inputs completes all the requirements for the execution of the decision. This makes the decision service available for execution by any participant that knows the decision URL.

The participants can optionally read the inputs before executing the decision. However not every participant is allowed to change the input. This would lead to unintended paths in the choreography. The participants responsible for editing the input can be identified in the choreography diagram because the input data originates from them. For example, the choreography diagram in Fig. 1 shows that the manufacturer announces the scores and, hence, is responsible for the creating or editing the input data used in the decision. When a request is sent for the execution of the decision, the response provides the output of the decision. The output of the decision should be consistent with the conditional sequence flows that follow the exclusive gateway.

### 3.2 Decision Services in RESTful Choreographies

Having a REST interface for the decision service is not enough. The objective is to go from a business process choreography to a RESTful choreography that solves the exclusive gateway problem by incorporating the RESTful decision service. In this subsection, we provide a stepwise method for embedding the decision service into the RESTful choreography. We assume that the initial business process choreography is syntactically correct. Since RESTful choreography is an extension of business process choreography, we do not want to introduce errors that break the correctness of the choreography. To this purpose, we consider two main properties: *The activity sequencing property* [1] The initiator of each choreography task is either initiator or recipient in the direct preceding task. The very first choreography task is an exception since it does not have any preceding task; *Hyperlink completeness* [9] All REST-request links used in the RESTful choreography are provided to the initiator at some point upstream in the choreography. An exception is made for the first occurring REST-request.

The decision service can be an external participant i.e., an additional business actor who provides decision services or hosted by one of the participants. In terms of the REST interface, that is irrelevant because only the web domain would be different and the rest of the URI would not change. For the remainder of this paper, we assume the more complex and interesting case where the decision server is an external participant in the RESTful choreography.

The stepwise method for embedding the RESTful decision service in business process choreography is given below. As mentioned above, the only requirement to start this method is the creation of the decision by the decision owner. The method consists of the following five steps:

1. Locate the choreography task where the decision-relevant data is passed for the first time. Add before the located task a new RESTful task that creates a new decision instance. The response should contain the location of the instance, the URL of the decision input, and the URL of the decision execution. The initiator should be the decision owner.
2. Next to the newly added task, add a new RESTful task that inserts the decision inputs using the link passed from the previous task. Again, the initiator should be the decision owner.
3. Change the content of the initiating message of every choreography task that models the passing of the decision-relevant data. Replace the content with information that describes the location of the decision together with the inputs and execution links.
4. Locate the exclusive gateway. Add before the gateway as many RESTful task as participants (affected by the gateway) in parallel. Each participant sends a request for executing the decision to the decision service and receives the output of the decision service as a response. This step can be put into a sub-choreography to not overload the diagram when we have more than two participants.

Now we analyse whether the activity sequencing properties is preserved after each step: Step 1. The insertion of the new RESTful task does not break the

property because the initiator remains the same; Step 2. The subsequent RESTful task added has again the same initiator. This means that the property is preserved because the initiator is present in the direct preceding task; Step 3. In this step, there is no change in the participants of each choreography task. Only the message content has been changed; Step 4. The BPMN specification requires that the initiators of the choreography task following the gateway have to part of the choreography task that directly precedes the gateway. Adding parallel REST calls that execute the decision service does not introduce any breach of the property because the initiators involved in the call are those affected by the gateway. As a conclusion, the transition from a choreography diagram to a RESTful choreography diagram (hosting a decision service) does not break the activity sequencing property of the diagram.

Regarding the hyperlink completeness property, we show that it is preserved. The link used for the first time in step 1 is the first occurring link. The concrete decision link, inputs link and execution link are provided to the decision owner with the response from the REST call of step 1. Then, the decision owner enters a new input using the inputs link in step 2. In step 3, all links introduced before are provided to the participants. Eventually, the participants use these links to execute the decision in step 5. Concluding, all the links (except the very first) are provided to the participants before being used by them.

Figure 3 shows the output of our overall approach. It implements unambiguously the parts procurements decision (modeled in Fig. 2) so that all suppliers take the correct path following the exclusive split gateway.

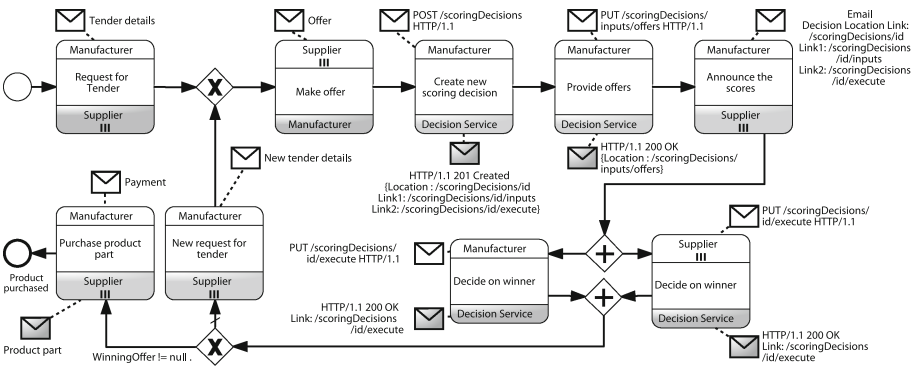


**Fig. 3.** A business process choreography for parts procurement by a manufacturer with the assist of a RESTful decision service

## 4 Conclusions

This paper tackles a problem related to the implementation of choreographies' exclusive gateways. We discuss the limitations of the BPMN specification regarding the implementation of choreography's exclusive gateways and state the problem induced by these limitations. Once the problem has been clearly stated,

we suggest a solution by introducing the RESTful decision service—a service that is based on the Decision Model and Notation standard.

The RESTful decision service provides a REST interface to allow the proper interaction of the participants with the decision service. Additionally, we provide a stepwise method for embedding such a service in any choreography diagram. The result is a RESTful choreography diagram that describes the complete interaction. The RESTful decision service makes the decision-relevant data visible and consistent assuring that decision is always executed on the same input data. Using DMN standard mitigates the misinterpretation of the decision output. Moreover, using this standard we achieve an increasing degree of separation of concerns between decision and process logic, hence, improving the choreography's comprehensibility and maintainability.

Finally, we illustrated our approach with an example which represents the interaction of a single business actor with an undefined number of business partners. Future work will look at more complicated interaction patterns with more complex decisions, where we believe that the RESTful decision service will play even a bigger role for facilitating the implementation of decisions in choreographies.

## References

1. OMG: Business Process Model and Notation (BPMN), Version 2.0, January 2011. http://www.omg.org/spec/BPMN/2.0/
2. Weske, M.: Business Process Management - Concepts, Languages, Architectures, 2nd edn. Springer, Heidelberg (2012)
3. OMG: Decision Model and Notation, Version 1.0, September 2015
4. Fielding, R.T.: Architectural styles and the design of network-based software architectures. Ph.D. thesis AAI9980887 (2000)
5. Nikaj, A., Mandal, S., Pautasso, C., Weske, M.: From choreography diagrams to RESTful interactions. In: Norta, A., Gaaloul, W., Gangadharan, G.R., Dam, H.K. (eds.) ICSOC 2015 Workshops. LNCS, vol. 9586, pp. 3–14. Springer, Heidelberg (2016)
6. Batoulis, K., Meyer, A., Bazhenova, E., Decker, G., Weske, M.: Extracting decision logic from process models. In: Zdravkovic, J., Kirikova, M., Johannesson, P. (eds.) CAiSE 2015. LNCS, vol. 9097, pp. 349–366. Springer, Heidelberg (2015)
7. Masse, M.: REST API Design Rulebook. O'Reilly Media Inc., Sebastopol (2011)
8. Palma, F., Gonzalez-Huerta, J., Moha, N., Guéhéneuc, Y.G., Tremblay, G.: Are RESTful APIs well-designed? Detection of their linguistic (anti)patterns. In: Barros, A., Grigori, D., Narendra, N.C., Dam, H.K. (eds.) ICSOC 2015. LNCS, vol. 9435, pp. 171–187. Springer, Heidelberg (2015)
9. Nikaj, A., Weske, M.: Formal specification of RESTful choreography properties. In: Bozzon, A., Cudré-Mauroux, P., Pautasso, C. (eds.) ICWE 2016. LNCS, vol. 9671, pp. 365–372. Springer, Heidelberg (2016). doi:10.1007/978-3-319-38791-8_21