Software & Systems Modeling (2019) 18:1195–1208
https://doi.org/10.1007/s10270-017-0653-2

SPECIAL SECTION PAPER

Semi-automatic derivation of RESTful choreographies from business process choreographies

Adriatik Nikaj[1] · Mathias Weske[1] · Jan Mendling[2]

Received: 20 December 2016 / Revised: 7 December 2017 / Accepted: 20 December 2017 / Published online: 2 January 2018

Abstract
Enterprises reach out for collaborations with other organizations in order to offer complex products and services to the market. Such collaboration and coordination between different organizations, for a good share, is facilitated by information technology. The BPMN process choreography is a modeling language for specifying the exchange of information and services between different organizations at the business level. Recently, there is a surging use of the REST architectural style for the provisioning of services on the web, but few systematic engineering approach to design their collaboration. In this paper, we address this gap in a comprehensive way by defining a semi-automatic method for the derivation of RESTful choreographies from process choreographies. The method is based on natural language analysis techniques to derive interactions from the textual information in process choreographies. The proposed method is evaluated in terms of effectiveness resulting in the intervention of a web engineer in only about 10% of all generated RESTful interactions.

Keywords Business process choreographies · RESTful choreographies · Natural language analysis

1 Introduction

Traditionally, research in BPM has focused on internal processes of organizations. The trend toward more complex services extends BPM toward a view of interactions between multiple processes. Such interactions, enabled by information technology, require standard models, like BPMN [1], which can be understood by all the participants. In particular, BPMN business process choreography specifies the interactions between two or more participants and the order in which these interactions take place at the business level. On the technical level, REST [2] is increasingly becoming the architectural style of choice for providing services on the web leading to the mainstream development of RESTful APIs.

The design of the RESTful APIs, which are involved in the concrete web interactions, should be based upon the business interactions modeled via the choreography diagram. However, taking business interactions down to the level of RESTful interactions is challenging. The designers of choreography diagrams are usually business-process domain experts and do not have knowledge of software development. The same holds for web engineers with respect to the business process choreographies, e.g., they are often too close to the implementation perspective when designing URIs, which makes the URIs harder to understand for the RESTful API's clients [3]. First concepts toward addressing this problem have been described in [4], however, with limitations in conceptual score and evaluation.

In this paper, we address this research gap in a comprehensive way and present an approach that takes as input a standard BPMN choreography diagram and it generates as output a RESTful choreography [5]. RESTful choreography is a useful [6,7] BPMN-based modeling language tailored to the specification of RESTful interactions between different business actors. Our approach is based on natural language processing techniques, which use textual descriptions of the choreography task to map to the most suitable

Communicated by Dr. Ilia Bider and Rainer Schmidt.

✉ Adriatik Nikaj
adriatik.nikaj@hpi.de

Mathias Weske
mathias.weske@hpi.de

Jan Mendling
jan.mendling@wu.ac.at

1 Hasso Plattner Institute, University of Potsdam, Potsdam, Germany

2 Institute for Information Business, WU Vienna, Vienna, Austria

REST verb with a corresponding REST URI. We implemented our approach in a research prototype and applied it on a set of choreography diagrams from different domains. The derived REST requests have also been evaluated by REST experts confirming the usefulness of our approach. The created RESTful choreography is used to derive code skeletons which facilitate the development of REST APIs. Applying our approach means that the same REST interface generation logic is used across all participants contributing to a better understandability, maintenance and evolution of the participants' RESTful APIs.

This approach extends the work in [4] by providing an enhanced label analysis approach and an extended evaluation in terms of a larger evaluation data set. This extension leads to a higher effectiveness in the correct generation of RESTful interactions, where a web engineer intervention is required in only about 10% of the generations.

This paper is organized as follows. Section 2 introduces choreography diagrams and the RESTful choreography diagram. These concepts are illustrated by a running example. Section 3 presents our semi-automatic approach of deriving RESTful choreography diagrams. Section 4 discusses the setup and the results of our user evaluation. Section 5 provides the related work before Sect. 6 concludes the paper and describes future work.

2 Preliminaries

This section briefly describes choreography diagrams by the help of an example. Additionally, REST architectural style is explained before the concept of RESTful choreography diagram is introduced.

2.1 Choreography diagram

The business process choreography diagram introduced in BPMN 2.0 [1] is a modeling language that focuses on the specification of the interactions between two or more participants, who, in general, are business actors, e.g., enterprises, customers, or organizations. Compared to business process models, the choreography diagram abstracts from the participants' internal processes and specifies the order in which the messages are exchanged between the participants.

Figure 1 depicts an example of a choreography diagram. This diagram describes the interaction between different participants involved in the submission, review, and organization processes with the goal of arranging a scientific conference. Some of the main stakeholders in a conference include the organizers, authors, and reviewers. The diagram depicts the interactions between these three participants starting from issuing a call-for-papers (CFP) and ending, in the best case, with the confirmation of the paper publication.

To facilitate these interactions, the participants make use of a Review Management System (RMS) which, in our case, is inspired by http://easychair.org. The RMS is responsible for coordinating these three participants throughout the entire collaboration.

As it can be seen in Fig. 1, the main element of a choreography diagram is the choreography task (graphically depicted as a rounded rectangle). It represents message exchanges between two participants. The participant initiating the message exchange is called the initiator, while the other participant is called the recipient. The return message is optional and can be sent from the recipient to the initiator. To graphically distinguish the initiator from the recipient, the latter is always highlighted in gray. The same applies for the initiating and return messages, although the messages are not required to be graphically depicted.

Choreography diagrams define the order in which the interactions are carried out. Choreography tasks have an order dependency that is modeled via sequence flows. The sequence flows, events and gateways are used in a similar fashion as in regular BPMN business process models. However, only a subset of events and gateways can be used in the choreography diagram. The events in Fig. 1 are the start event, timer event *deadline* and the three end events: *Paper rejected*; *Short paper declined*; *Paper accepted for publication*.

2.2 RESTful choreography diagram

The REST architectural style [2] is increasingly used for the development of RESTful web services. Its architectural constrains contribute to among others, better scalability and portability. In virtually all cases, REST uses the HTTP protocol as a means of interactions between different participants. The interaction is achieved by using standard HTTP verbs (GET, POST, PUT, DELETE) on resources. The resources resting on the server are globally and uniquely identified via URL. Their state can be changed by the client through these REST verbs. Due to the stateless constraint, the server does not need to remember previous interactions with the client in order to understand the client's request.

The assumption of the RMS example is that all participants are RESTful services, i.e., they interact with each other by sending REST calls. In simple browser settings, the organizer, reviewer and author are users of RMS. In this example, we assume that they also provide a RESTful API. Consider these services as RMS mobile applications where RMS can push notifications [8] depending on the user role, e.g., notifying the reviewers about the papers assigned for review, sending the paper decision to the authors, informing the organizer when all reviews are submitted.

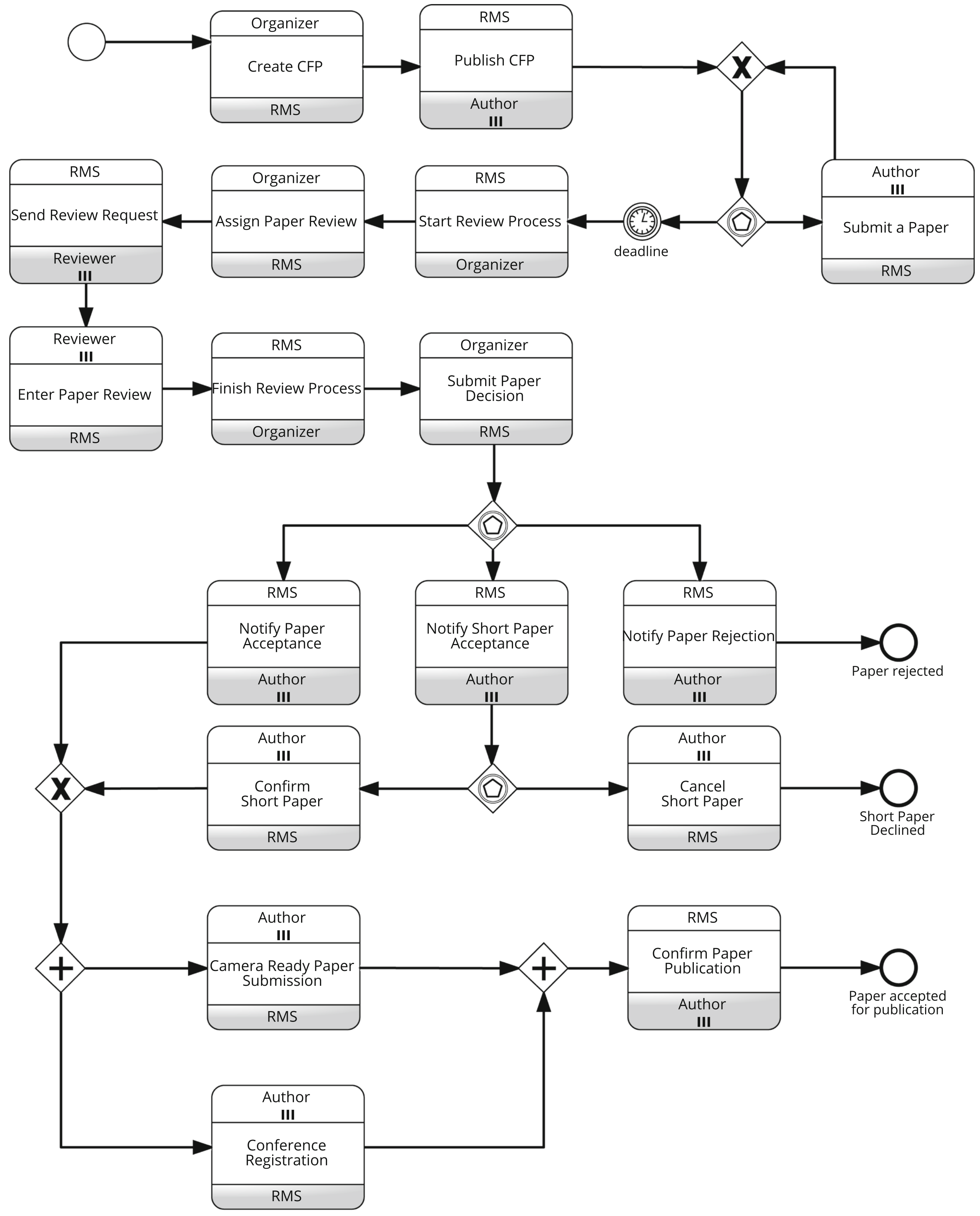


Fig. 1 Choreography diagram for paper submission and review management

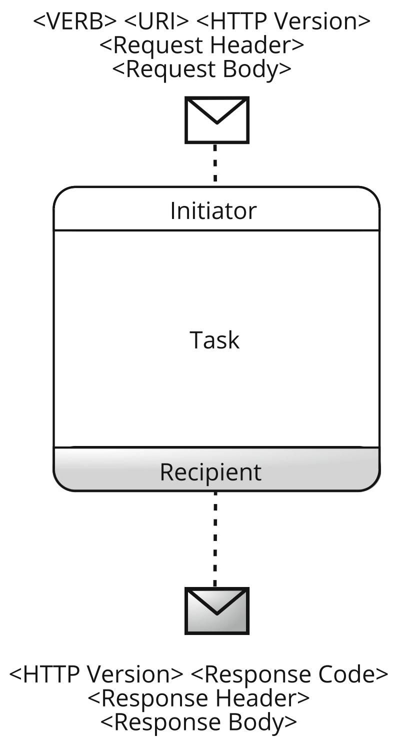


Fig. 2 The annotation of the choreograph task in RESTful choreography [5]

Business processes can be used to model the internal behavior of the participants involved in RESTful conversation as proposed in [9]. However, when it comes to interactions between multiple participants, it is important to focus on a global perspective in order to capture the state of common resources and the allowed interactions with these resources.

To this end, Nikaj et al. [5] introduce RESTful choreography diagrams—a lightweight enhancement of BPMN choreography diagram with REST details. These details include annotations for the choreography tasks that represent a RESTful interaction, called a RESTful task, like the *Submit paper decision* choreography task in Fig. 1. This is realized by refining the two messages of choreography task, respectively, into a REST request and a REST response like depicted in Fig. 2. Figure 3 provides an excerpt of a RESTful choreography diagram modeling the RESTful interaction between the organizer, RMS and the author. However, the person responsible for the enhancement of the choreography task with REST notations has to understand both the business aspect of the choreography and the implementation aspect of the RESTful interaction. This problem is addressed in our paper by proposing a semi-automatic approach for deriving RESTful choreographies from business process choreographies.

3 Semi-automatic generation of RESTful choreography

This section presents our semi-automatic approach to generate RESTful choreographies. Section 3.1 discusses the relevant concepts of the approach. Section 3.2 expends on the previous section focusing on choreography-specific labeling style. Section 3.3 then explains how the concepts are employed to identify the type of REST request that is

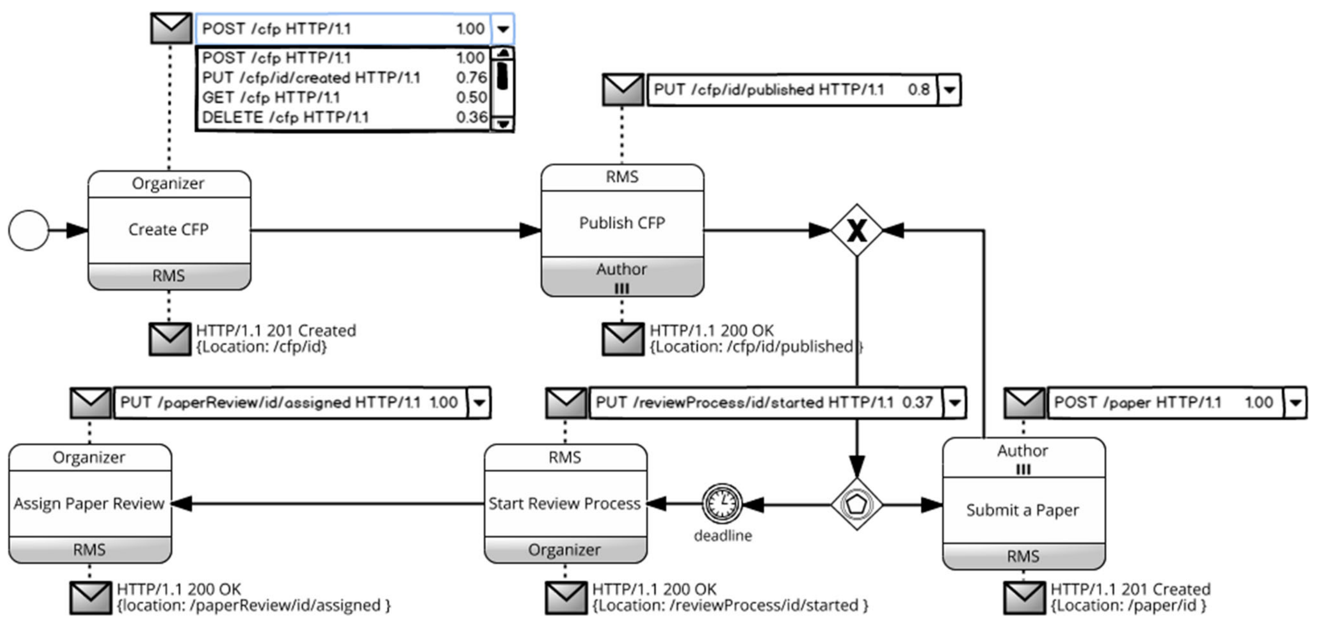


Fig. 3 A part of the generated RESTful choreography of RMS

expressed in a choreography task label. Section 3.4 shows how choreography tasks are finally enriched with RESTful information.

3.1 Foundations

This subsection starts with a formal specification of a choreography diagram as the core artifact of our approach. We use all the elements of the choreography diagram that are needed to generate a RESTful choreography diagram. We consider a choreography diagram to be a tuple $C = (N, S, P, L, label)$, such that:

- $N = T \cup E \cup G$ is a set of nodes. T, E and G are pairwise mutually exclusive;
- T is a set of choreography tasks;
- E is a set of events;
- G is a set of gateways;
- $S \subseteq N \times N$ is a set of sequence flows;
- P is a set of participants;
- L is a set of natural language text labels;
- $label : T \mapsto L$ is a function which assigns a text label to a choreography task.

As pointed out in Sect. 2, a choreography task can represent either a single message exchange or two message exchanges (i.e., a send message and a reply message). In our approach, we map each case to a single RESTful interaction in that the initiator makes a REST request to the recipient. Since the optional second message, according to the BPMN specification [1], is a return message, we do not consider it as a new REST request but rather a response from the recipient (the server in REST terms) which is embedded in the HTTP response body (see Fig. 2). Additionally, we observe that the choreography task label, which is subject to natural language processing, is more similar to its corresponding business process send task than receive task. This means that the choreography task label provides no information on how to name the second REST request.

In order to process the textual information of the labels, it is necessary to access this information in a structured way. As a starting point we observe, as mentioned above, that choreography tasks are similarly labeled as activities, often referring to the corresponding send task in a business process model [1]. Thus, we assume that each label of a choreography task contains the following components: an action and a business object on which the action is applied [10]. As an example, consider the label *Submit paper decision* from Fig. 1. It contains the action *to submit* and the business object *paper decision*. It is important to note that these components can be communicated in different grammatical variations. For instance, the label *camera ready paper submission* communicates the action in a different grammatical structure by using nouns, which express the action *to submit*.

In order to be independent of grammatical labeling structures, we rely on the label annotation approach of Leopold et al. [11] which identifies actions and business objects with a decent degree of accuracy (Avg. precision: 91%, avg. recall: 90.6%). Considering $l = label(t) \in L$ to be the label of an arbitrary choreography task and considering W_V and W_N to describe the set of all verbs and nouns, respectively, we refer to the action and the business object of l as follows:

- $\alpha : L \mapsto W_V$ is a function that assigns an action to a choreography task label;
- $\beta : L \mapsto W_N$ is a function that assigns a business object to a choreography task label.

As an example, consider the choreography task labeled with *Submit paper decision* from Fig. 1. According to the prior conceptualization, the action is given by α (Submit paper decision) = *to submit* and the business object is given by β (Submit paper decision) = *paper decision*. We will use these label components in the following to derive the respective REST requests and to generate the RESTful annotations from the text labels of choreography tasks.

3.2 Choreography-specific label analysis

In the aforementioned label processing approach, we observed that the assumption about the similarity of choreography task and business process task does not always hold. Since choreography tasks represent a message which is sent across organizations, the labels tend to contain the word *send* or a similar one, like *enter*, *submit*, *mail*, which describes the passing of a message from one choreography participant to another. These words are followed by single or multiple words, e.g., *order*, *invoice*, *request*, *application request*, *paper submission*, *short paper acceptance*. The word *send* indicates no semantic preference toward any specific REST verbs since all REST requests are sent messages. For example, one can send a GET request for retrieving a resource representation or a DELETE request for deleting a resource. Therefore, we turn our focus on the remainder of the label containing the word *send* (or a similar one) to look for some additional semantic information which can indicate an inclination of the label toward a specific REST verb.

Before checking whether the label contains the action *send* or the like we need to identify the verbs which can be used in a choreography just to represent a message being sent. Synonym and semantically similar verbs of *send* are provided in Table 1. The way we populate this set is explained in the next section.

If the label has an action that is element of Syn_{Send}, we analyze the remainder of the label. We reapply the label anno-

Table 1 Synonym word sets of the REST verbs and *send*

Verb	Description	Synonym word sets
POST	Creation of a new resource on the server	Syn_{POST} = {create, request, [produce, make, …] }
PUT	Editing an existing resource	Syn_{PUT} = {confirm, edit, accept, send [support, redact, …] }
GET	Retrieving an existing resource from the server	Syn_{GET} = {retrieve, read, [get, find, recover, …] }
DELETE	Deleting an existing resource	Syn_{DELETE} = {cancel, delete, [erase, postpone, …] }
Send	Sending a message from one choreography participant to another	Syn_{Send} = {"send", "enter", "submit", "notify" [mail, transmit, …] }

tator (from previous subsection) on the remainder to identify a new action and business object. For example, the label *send review request*, *notify paper rejection*, *notify short paper acceptance* are reanalyzed as *review request*, *paper rejection* and *short paper acceptance*. Applying the same annotation approach would yield the following action–business object pairs: *request-review*; *reject-paper*; *accept-short paper*.

If, however, there is no new action and business object, we treat this label in the same way we would do with any other verb. For example in a label *send review*, we have α(send review) = *to send* and β (send review) = *review*. In this case, we map *send* to the REST verb PUT to express that the sent resource is in the state "sent" (see Sect. 3.4). The remainder does not need to be a single word, e.g., in the label *send short paper*, analyzing the remainder *short paper* would yield no new business object. Again in this case, we map *send* to the REST verb PUT.

The remainder analysis provides more semantic information which is needed to generate the REST request as described in the following two sections.

3.3 REST verb derivation via natural language analysis

The general idea of deriving REST requests via natural language analysis is based on the assumption that the choreography task label provides all relevant information. Specifically, we focus on the actions of the labels since they describe which specific activities have to be carried out and how these activities affect the system. The REST verb derivation applies two steps. The first step compares the action of the respective choreography task label with synonym words that reflect the meaning of the different REST verbs. The second step involves a linguistic similarity analysis of the action of the choreography task label and the synonym words, in case the action of the label does not exactly match with any of the synonym words. In the following, we discuss these two steps in further detail.

First, we require a set of synonym words before we can conduct the derivation. A challenge is that the REST verbs are associated with a specific technical meaning that does not necessarily correspond with the linguistic meaning of a word. For example, the REST verb *POST* instructs the server to create a new distinguishable resource, while the verb *to post* typically describes the act of publicizing news on bulletin boards. Therefore, it is necessary to define a set of synonym words that reflect the meaning of *POST* in a technical sense. For this purpose, we asked REST experts for natural language verbs that best resemble the meaning of the REST verbs. The result of this process is shown in Table 1. For example, the experts agreed that the meaning of *POST* is best reflected by the verbs *to create* or *to request*. As the identified verbs might not capture all the variation in language, we further consider additional synonyms that may be extracted from computational lexicons, such as WordNet [12]. For example, a *POST* verb might also be related to the verbs *to produce* or *to make*. Other examples may be retrieved from the previous table in edged brackets. Similarly, we use the same approach to derive the *send* synonym set in Table 1.

The *synonym analysis step* investigates whether or not the action of a choreography task label equals one of the synonym words of the REST verbs. If this condition evaluates to true, we map the respective REST verb to the choreography task. Otherwise, no REST verb is mapped to this task. As an example, consider the choreography tasks *Create CFP* and *Confirm Short Paper*. The first task would map to *POST* because its action *to create* is a member of the synonym words of the set Syn_{POST}. The second task would map to *PUT* since its action *to confirm* is a member of the set Syn_{PUT}. This logic is expressed by the following function, assuming *REST* to be the set of REST verbs:

$$syn(l) = \begin{cases} POST & \text{, if } \alpha(l) \in Syn_{POST} \\ PUT & \text{, if } \alpha(l) \in Syn_{PUT} \\ GET & \text{, if } \alpha(l) \in Syn_{GET} \\ DELETE & \text{, if } \alpha(l) \in Syn_{DELETE} \\ \emptyset & \text{, otherwise} \end{cases} \quad (1)$$

The *similarity analysis step* serves as a fallback strategy in case the synonym analysis step fails to assign a REST verb to a choreography task. In this case, it is necessary to find a REST verb that is most closely related to the action. Therefore, it is necessary to determine the relatedness of an action with the synonym words. In our approach, we use the notion of semantic similarity (see, e.g., [13–15]) to quantify this relatedness. We utilize the distributional similarity of the DISCO word similarity tool [16], denoted with sim_{DISCO}, because it outperforms existing similarity measures [17]. Given a choreography task label l, its action $\alpha(l)$, and the set of synonym words of an arbitrary REST verb Syn_{REST}, the relatedness of an action of a choreography task label and a synonym REST set is given as follows:

$$rel(\alpha(l), Syn_{REST}) = \max_{w \in Syn_{REST}} sim_{DISCO}(\alpha(l), w) \quad (2)$$

As an example, we consider the choreography task *Enter paper review* from Fig. 1. Since the action *to enter* is not member of the synonym sets of the REST verbs, we determine its relatedness to each synonym set. Using the second-order distributional similarity, we receive the following relatedness values: rel(enter, Syn_{POST}) = 0.48, rel(enter, Syn_{PUT}) = 0.92, rel(enter, Syn_{GET}) = 0.92, rel(enter, Syn_{DELETE}) = 0.55.

Finally, we consider all of the relatedness scores to derive the most suitable REST verb for a given choreography task label. In this case, we assume that the highest relatedness score reflects the most suitable REST verb for a given choreography task. Accordingly, we assign this REST verb to the highest relatedness score. However, it might be the case that several relatedness scores are equal which consequently leads to more than one assignment of a REST verb emphasizing the necessity of a user to choose the correct REST verb. Formally, we describe the similarity analysis step as follows:

$$sim(l) = \{r \in REST | \max rel(\alpha(l), Syn_r)\} \quad (3)$$

As an example, consider again the choreography task *Enter paper review* and its relatedness scores. Since the scores of PUT and GET are equal, the similarity analysis strategy assigns both REST verbs PUT and GET to the choreography task.

The label's action is not the only entity which can define the appropriate REST verb. We take in consideration also the identified business object. The difference between PUT and POST in a REST call consists in the different ways the server treats the resource. When using a POST, the server creates a resource assigning the identifier on its own. However, when using PUT, the client has to specify the identifier of the resource in the request. For example, the request POST /paper is followed by a response /papers/id where the id is chosen by the server. Otherwise, the author should know the id of the paper he or she wants to modify or create when he or she requests PUT /paper/id.

We observed that this inherent difference between POST and PUT can be found in the natural language notion of indefinite and definite article, respectively. In those cases the action is mapped to PUT, we perform an additional check for the presence of the English indefinite article "a" or "an". We then map the action to POST if that turns to be the case. For example, *submit a paper* task from Fig. 1 is mapped to POST /paper since the client does not refer to a specific paper.

The following section will explain how RESTful requests are generated for the choreography tasks using the respective identified REST verb.

3.4 REST request generation

The task of generating REST requests involves the generation of a unique resource identifier (URI) explaining how the resource is addressed via the HTTP. In order to generate a URI, we consider its generation as a language generation problem that uses the available information of the choreography task and the REST verb derivation from the previous step. Many language generation systems take a three-step pipeline approach that first determines the required information of a sentence, second plans the expression of this information, and third transforms them into correct sentences [18]. In contrast to these systems, we do not require a fully flexible approach, since the final links follow regular structures [5]. Therefore, we use a template-based approach [19–21] to generate REST URIs. In particular, we use the choreography task together with the REST verb from the previous step and select the respective link template. Afterward, we fill the template with the necessary information, i.e., action and business object of a choreography task label. It has to be noted that there are cases in which a choreography is not associated with another REST link when the request derivation reveals more than one or no REST verbs requiring the user to correct the links.

Table 2 shows link templates for the different REST verbs and gives examples created from the choreography tasks of Fig. 1. The templates emphasize that the business object of a choreography task label ($\beta(l)$) plays an important role for the REST links since it resembles the server resource that needs to be addressed by a REST verb. We therefore associate the business object together with a unique identifier. In case the state of a specific resource has to be changed, the link also explains how its state changes with the REST verb. This

Table 2 Link templates for REST requests based on [5]

Link template	Example
POST /< $\beta(l)$> <HTTP Version>	POST /CFP HTTP/1.1
PUT /< $\beta(l)$ >/id/<Past Participle of α > <HTTP Version>	PUT /paperReview/id/entered HTTP/1.1
GET /< $\beta(l)$ >/id <HTTP Version>	GET /paperReview/id HTTP/1.1
DELETE /< $\beta(l)$ >/id <HTTP Version>	DELETE /shortPaper/id HTTP/1.1

change is expressed by the past participle of the action of a choreography task label.

The final output of our approach is a RESTful choreography. Figure 3 is a mock-up that depicts an excerpt of the RESTful choreography diagram generated by applying our approach to the running example. In this figure, we show how the REST engineer can interact with the generated RESTful choreography. The REST engineer is provided with all four generated links (one for each REST verb) ranked based on the matchmaking score (1 being the best and 0 the worse). Depending on the selection the HTTP response is generated automatically, assuming that the interaction is always valid.

4 Evaluation

This section describes our evaluation. First, we explain the architecture of our prototypical implementation. Then, we present results on the accuracy of the derivation steps for a set of 172 choreography diagrams from practice.

4.1 Evaluation setup

For evaluating our approach, we developed a tool, called REST Annotator. The architecture of the REST Annotator is depicted in Fig. 4 as an FMC diagram [22]. The REST Annotator takes a set of choreography diagrams as an input and it outputs a set of REST-enriched choreography diagrams. The tool makes use of three external components: the label annotator by Leopold et al. [11], WordNet [12], and the distributional similarity component of the DISCO tool [16]. The main component constituting the tool is composed of three sub-components: Label Analyzer, REST verb identifier and REST URI identifier.

The Label Analyzer is responsible for extracting all the labels from the model and analyzing them with the help of label annotator. The latter is used to notate the action and the business object of a choreography task label. The Label Analyzer maps the action and the business object for each label to the REST Verb Identifier and the REST URI Identifier components. The REST Verb Identifier component requires the action provided by the Label Analyzer and the synonyms of WordNet resembling the respective REST verb. If no synonym is found, the component requires the semantic similarity score between the action and the synonym sets of the REST verbs from the Disco Semantic Similarity component. Once the semantic relation of the action with each of the REST verbs is identified, the REST verb and its respective score is passed to the REST URI Identifier component. This component generates as outputs the

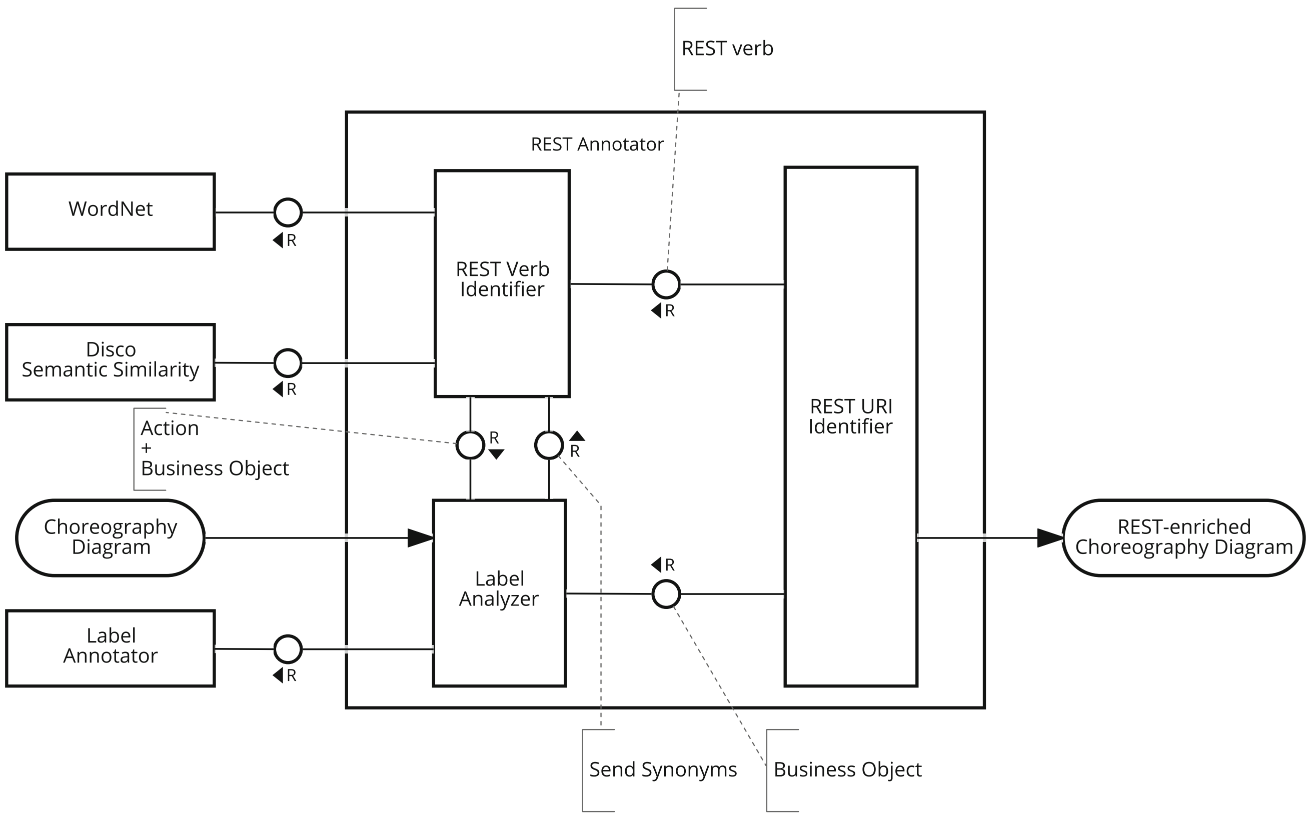


Fig. 4 REST Annotator Software Architecture

set of choreography diagrams enriched with REST annotation.

Additionally, the Label Analyzer, before providing the final action and business object, needs to apply the approach described in Sect. 3.2. To this end, it requires Syn_{Send}, which is generated by the REST Verb Identifier. In addition, the latter needs from the former the business object to check for the presence of an indefinite article. As explained in Sect. 3.3, the presence of an indefinite article can alter the output of the REST Verb Identifier.

As evaluation data, we use choreography diagrams from the BPM Academic Initiative. The initiative offers a rich set of process models from different domains. Overall, we retrieve 424 BPMN choreography diagrams. Since these diagrams are created from experts and non-experts alike, it is necessary to *clean the data*. We apply the following cleaning criteria:

1. *English-only diagrams* We include only diagrams with English text labels. This criterion is necessary because most of the natural language analysis components only support English.
2. *Syntactically correct diagrams* Diagrams which have syntax errors with respect to the BPMN 2.0 choreography diagram specification are excluded.

With regard to the *evaluation procedure* a REST expert had to perform a three-step evaluation for each choreography task: (a) the syntax correctness of a label, (b) the adequate generation of the REST verb, and (c) the suitability of the generated REST URI. In case (a) holds true, the evaluator further has to check if the identified REST verb with the best matching score is adequate. In case (b) holds, the evaluator has to check if the generated URI is suitable.

The evaluation comes, however, with its own limitations. The existence of a single evaluator may impose subjectivity in determining the correct match. To mitigate this issue, the evaluator repeated several time the inspection procedure while consulting common practices in developing RESTful APIs. This avoids errors like inconsistent verification for the same label—RESTful request pairs—and helps to resolve the non-obvious pairs.

4.2 Evaluation results

This section discusses the results which are summarized in Table 3. The 172 models contain 1213 choreography task labels in total. The models size (in terms of choreography task count) ranges from 1 to 26 with an average of about 7 choreography task per model. From these labels, 864 labels (71.14%) are syntactically correct labels. In the following discussion, we only focus on those labels that are syntactically correct and discuss how the verb identification and the link generation performs in these cases.

Table 3 Quantitative results of the user evaluation

Total no. of labels	1213
No. of syntactically correct labels	864 (71.46%)
No. of syntactically incorrect labels	349
Total no. of correctly identified REST verbs	772 (89.35%)
.. with the synonym identification strategy	322
.. with the similarity identification strategy	450
.. POST	139 (97.89%)
.. PUT	577 (88.36%)
.. GET	54 (80.60%)
.. DELETE	2 (100.00%)
Total no. of incorrectly identified REST verbs	92 (10.65%)
Total no. of correct links	723 (93.65%)
Total no. of incorrect links	49 (6.35%)

The *verb identification strategies* have identified the correct REST verb in 772 labels which amounts to 89.35% of all syntactically correct labels. Among these labels, we further distinguish between the verbs that have been identified with the synonym strategy and the similarity strategy. The synonym strategy is capable of deriving the correct REST verb in 332 labels, while the similarity strategy derives the correct REST verb for 450 choreography labels. The results emphasize the need for the similarity identification strategy of the REST verb. The most identified REST verb is PUT as it is expected in a choreography context where participants change the state of business objects, e.g., order is sent, accepted, delivered and payed. POST was identified in 139 cases, 8 of which were identified using the presence of the indefinite article. The low number of DELETE identifications reflects also the rare cases of using DELETE in the REST context due to resources being often archived or saved in a particular state rather than being deliberately deleted.

In total, 92 choreography labels (10.65%) have been annotated with the wrong REST verb. We observe that GET is detected the least and DELETE is always detected. We identify two classes of errors that can lead to the wrong annotation, the first of which is fixed in the context of this sample set and does not count toward the incorrect REST verb identifications. This first class subsumes choreography labels for which the similarity strategy revealed two or more equal similarity scores. This has been the case for 101 choreography labels. After identifying the list of these REST-ambiguous actions for this particular sample set, a REST expert was asked to choose the most appropriate mapping. The following non-exhaustive list is disambiguated: {*start-PUT, pay-PUT, invoice-PUT, article-*

PUT, enter-PUT, publish-PUT, allocate-PUT, explain-PUT, disburse-PUT, receipt-PUT, show-GET, book-PUT}. This list can be used and enriched further with verbs that score equally in the semantic similarity approach.

The second class covers such cases in which our approach identified the wrong verb. The REST evaluation has revealed 92 choreography labels for which our approach did not find the correct REST verb. These cases have to be corrected by the user.

The approach to *generate RESTful links* has created 723 correct and 49 incorrect links out of 772 correct verb identifications. We identified the labeling quality as a main cause for the incorrect links. For example, we found choreography tasks that have not been specified correctly by referring to a particular state, e.g., *payment confirmed* and *invoice sent* are mapped to PUT /payment/id/confirmeded HTTP/1.1 and PUT /invoice/id/sented HTTP/1.1, respectively. A correct result is generated for the labels *confirm payment/payment confirmation* and *send invoice*. Another cause for the incorrect link generations is the misidentification of the business object. For example, the label *ship article* is labeled as α(ship article) = *article* (action) and β(ship article) = *ship* (business object). The correct labeling would be to identify *ship* as the action and *article* as the business object. Nevertheless, we conclude that the link generation works satisfactory and produces a large number of correct REST links.

We also exemplify the results of our evaluation by applying our approach to the exemplary choreography diagram from Fig. 1. Table 4 shows the generated REST requests for the respective choreography tasks.

4.3 Discussion

Three main observations emerge from the quantitative evaluation results. The first observation relates to the correct annotation of choreography tasks with REST URIs. For example, it identifies *PUT* to be the correct REST verb for the task *confirm short paper* and generates the URI *PUT /shortPaper/id/confirmed*. However, we also encounter problems for cases, in which the approach retrieves several possibilities for REST verbs and fails to make a decision for one particular REST verb. In the example, the choreography task *enter paper review* falls into this group. The approach identifies the REST verbs *PUT* and *GET* because the action *to enter* is not a member of any REST verb synonym list and the semantic similarity score is equal for both REST verbs. Based on this result, the link generator component creates two possible links, among which the user has to choose. Nevertheless, the links themselves have been created correctly. As mentioned in the previous section, we solved this problem for this particular test set by disambiguating the REST verb mapping. However, the list of disambiguated verbs is not exhaustive at its current form because there are probably other verbs which were not part of the labels we used in our evaluation. The list can be used as input when applying our approach to achieve better results for choreography labels which contain such verbs.

The third observation covers REST requests that are incorrect and that need to be manually corrected by the user. As an example, consider the choreography task *conference registration*, for which our approach creates a *GET* link. However, we would expect a *POST* or a *PUT* request. Incorrect links

Table 4 REST request results for the corresponding RESTful tasks from Fig. 1

RESTful task	REST request
Create CFP	POST /cfp HTTP/1.1
Publish CFP	PUT /cfp/id/published HTTP/1.1
Submit a paper	POST /paper HTTP/1.1
Start review process	PUT /reviewProcess/id/started HTTP/1.1
Assign paper review	PUT /paperReview/id/assigned HTTP/1.1
Send review request	POST /review HTTP/1.1
Enter paper review	PUT /paperReview/id/entered HTTP/1.1
Finish review process	PUT /reviewProcess/id/finished HTTP/1.1
Submit paper decision	PUT /paper/id/decided HTTP/1.1
Notify paper rejection	PUT /paper/id/rejected HTTP/1.1
Notify short paper acceptance	PUT /shortPaper/id/accepted HTTP/1.1
Cancel short paper	DELETE /shortPaper/id HTTP/1.1
Confirm short paper	PUT /shortPaper/id/confirmed HTTP/1.1
Camera-ready paper submission	PUT /cameraReadyPaper/id/submitted HTTP/1.1
Conference registration	GET /conference HTTP/1.1
Confirm paper publication	PUT /paperPublication/id/confirmed HTTP/1.1

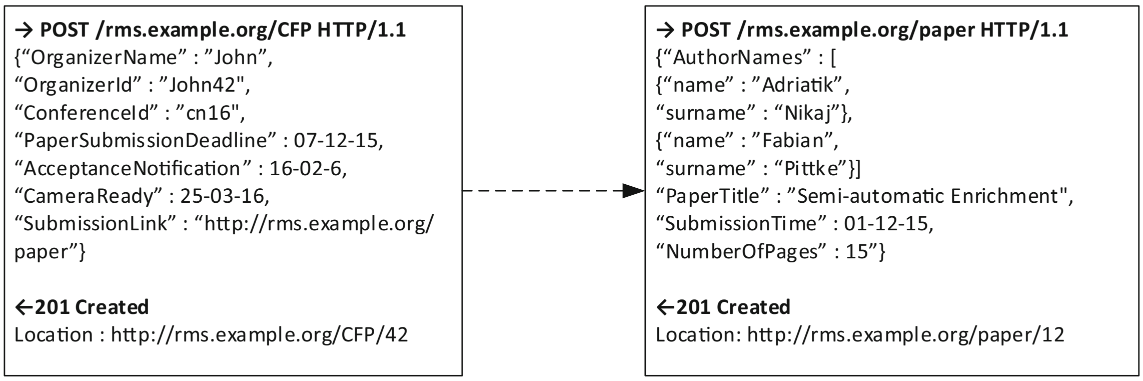
of this type may have several error sources. On the one hand, the Label Annotator component (see Fig. 4) might have misclassified the choreography task and erroneously changed action and business object. On the other hand, the REST verb identification component might have caused the error because the action is either a direct member of the synonym word lists or its similarity score with the synonym words is highest for one of the other REST verbs. In our example, the former applies. The REST verb *GET* has been identified, since the action *to register* is a WordNet synonym of *to read* and thus a member of the synonym word set Syn_{GET}. Hence, the other alternatives are not considered so far, which finally requires the user to correct this REST request.

At last, Fig. 5 depicts a concrete instance of the RMS RESTful interaction. The part in bold and the order of REST interactions are generated by the REST Annotator tool and provided to the developer as a skeleton to follow for developing the RESTful API. In the RSM context, the two rectangles represent, respectively, the concrete instances of the *create CFP* and *submit a paper* choreography tasks from Fig. 3. The dashed arrow expresses that the second instance can only be executed only after the first one is executed. For a given RESTful choreography, a skeleton diagram can be derived for each participant who offer a RESTful API, like the RMS mobile app of the conference organizer which receives notification from RMS about the status of the reviewing process . Hence, we jump from a global choreography view, to at least one orchestration view that focuses only on the REST behavioral interface i.e., the order in which the REST requests and responses are performed within a single participant application. The benefit of applying our approach is in that the same URI generation logic is used across all participants contributing to a better understandability, maintenance and evolution of REST APIs [23]. The automation of deriving skeletons from a RESTful choreography is left as a future work.

5 Related work

We identify three major groups of research related to our approach. First, our approach is related to model-driven approaches that focus on the process of designing and engineering REST APIs or RESTful services. Examples include the work from Valverde and Pastor [24] or Schreier [25], who support this process by providing metamodels. While the former metamodel focuses on the specification of REST services and the generation of machine-readable specifications, the latter approach addresses formal aspects of a REST application, such as application structure and behavior. Laikorpi et al. [26] consider the design of a RESTful API as a model transformation problem and describe necessary transformations and intermediate models for developing RESTful services. Our approach contributes to model-driven approaches by deriving REST information from choreography diagrams in a semi-automatic way. In contrast to these approaches, our approach is based on the BPMN choreography standard, which specifies business interactions from a global perspective to derive REST skeletons with implementation details.

Second, our approach relates to the idea of bridging the gap between the business process choreography with its underlying orchestration system. With this regard, Decker et al. [27] propose an extension of BPEL web service composition standard [28] for closing the gap between composition and choreographies. The aim of the BPEL4Chor extension is to orchestrate process choreographies by integrating existing BPEL service orchestrations. BPEL4Chor is a bottom-up approach and it is based on web services standards like SOAP and WSDL [29]. Opposite to that, we take a top-down approach for deriving RESTful interactions. Another approach establishes the relation between BPMN and REST [9]. The author suggests that a part of a business process, per se, can be published as a REST resource. While this approach focuses on the internal behavior of the participant involved in a RESTful interaction, we focus on the global perspec-



```
→ POST /rms.example.org/CFP HTTP/1.1
{“OrganizerName” : ”John”,
“OrganizerId” : ”John42",
“ConferenceId” : ”cn16",
“PaperSubmissionDeadline” : 07-12-15,
“AcceptanceNotification” : 16-02-6,
“CameraReady” : 25-03-16,
“SubmissionLink” : “http://rms.example.org/
paper”}

←201 Created
Location : http://rms.example.org/CFP/42
```

```
→ POST /rms.example.org/paper HTTP/1.1
{“AuthorNames” : [
{“name” : ”Adriatik”,
“surname” : “Nikaj”},
{“name” : ”Fabian”,
“surname” : “Pittke”}]
“PaperTitle” : ”Semi-automatic Enrichment",
“SubmissionTime” : 01-12-15,
“NumberOfPages” : 15”}

←201 Created
Location: http://rms.example.org/paper/12
```

Fig. 5 A concrete skeleton instance of RMS implementation

tive, which allows reasoning about the allowed interactions at the implementation level. Moreover, the added value of this work consists in providing a semi-automatic methodical approach to derive RESTful choreographies from original business process choreographies.

<>Third, our approach is part of a broader context of business process to execution transformations. In [30], Mendling et al. show how BPEL process definitions can be derived from a global WS-CDL [31] model for each participant of the choreography. Moreover, this derivation is fully automatized for certain blocks and semi-automatized for those blocks whose context plays an important role. Similarly, Ouyang et al. propose in [32] a set of techniques to translate BPMN models into BPEL. The automatic translation does not impose structural restriction on the source BPMN model, and the target model is readable BPEL code. Transformations to BPEL also exist from other graph-based process modeling languages [33,34]. In the same direction as with the previous work, Weber et al. present a new approach in [35] that makes use of novel blockchain technology to implement business process collaboration. Blockchain provides a global computation infrastructure which can run programs that are referred to as *smart contracts* [36]. In [35], smart contracts are derived from process specifications and deployed into block chains for executing the process collaboration. However, none of these approaches makes use of Natural Language Processing to derive execution artifacts. Therefore, this paper provides a unique contribution in this way.

6 Conclusions, limitations and future work

The paper defines a semi-automatic approach for deriving RESTful choreographies from BPMN choreography diagrams. The proposed approach is based on natural language analysis techniques to derive the most suitable REST verb for the interaction and to generate a REST URI for the derived REST verb. Choreography-specific labeling style is taken into account. Our approach was evaluated by developing the REST Annotator tool and applying it to choreography diagrams from different domains. The output of the tool was assessed by a REST expert. The verb identification is correct in 89.35% of cases, while the URI is correct in 93.65% of cases. This work contributes an additional step toward the research gap between business process choreographies and their implementation.

Our approach also has limitations, which are grounded in the imprecise nature of natural language and the capabilities of the employed language processing tools. This imprecision is an important cause for several incorrectly identified REST verbs and REST URIs, which have to be corrected by REST experts. A particular limitation is related to the labeling style. If too many nouns are used, it is hard to identify the intended action and business object. For example, the label *application letter submission* would yield *PUT letter/applied* instead of the more preferable *PUT applicationLetter/submitted*.

In future work, we plan to address these limitations by making use of word sense disambiguation technology and of behavioral aspects of the choreography diagram. Word sense disambiguation utilizes external knowledge repositories such as WordNet [12] or BabelNet [37] together with contextual information or speech acts [38,39] in order to identify the correct interpretation of a word. Its usefulness has already been investigated for process models in [40]. Behavioral aspects relate to the sequential order of choreography tasks [41]. The fact that only certain sequences and combinations of messages make sense can be used to describe constraints that restrict the number of potential interpretations [42]. For example, if a POST and a GET request have been identified and the respective choreography task is at the beginning of the interaction, then it is more likely to be a POST request. In this way, we aim to improve the accuracy of the proposed method. Furthermore, this approach does not consider messages and their labeling. Including them may result in an increase of the URI generation accuracy as the messages describe the business object passed to the recipient.

References

1. OMG: Business Process Model and Notation (BPMN), Version 2.0. http://www.omg.org/spec/BPMN/2.0/ (2011)
2. Fielding, R.T.: Architectural styles and the design of network-based software architectures. Ph.D. thesis (2000)
3. Massé, M.: REST API Design Rulebook. O'Reilly Media Inc., Newton (2012)
4. Nikaj, A., Pittke, F., Weske, M., Mendling, J.: Semi-automatic derivation of RESTful interactions from choreography diagrams. In: Schmidt, R., Guédria, W., Bider, I., Guerreiro, S. (eds) Enterprise, Business-Process and Information Systems Modeling: Proceedings of the 17th International Conference, BPMDS 2016, 21st International Conference, EMMSAD 2016, Held at CAiSE 2016, Ljubljana, Slovenia, June 13–14, pp. 141–156. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-39429-9_10
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) Service-Oriented Computing – ICSOC 2015 Workshops: WESOA, RMSOC, ISC, DISCO, WESE, BSCI, FORMOVES, Goa, India, Nov. 16-19, 2015, Revised Selected Papers, pp. 3–14. Springer, Berlin, Heidelberg (2016). https://doi.org/10.1007/978-3-662-50539-7_1
6. Nikaj, A., Batoulis, K., Weske, M.: Rest-enabled decision making in business process choreographies. In: International Conference on Service-Oriented Computing, pp. 547–554. Springer (2016)
7. Nikaj, A., Weske, M.: Formal Specification of RESTful Choreography Properties. In: 16th International Conference on Web Engineering, ICWE 2016, Lugano, Switzerland, June 6–9, 2016. Springer (2016)
8. Pautasso, C., Wilde, E.: Push-enabling RESTful business processes. In: Kappel, G., Maamar, Z., Motahari-Nezhad, H.R. (eds.) Service-Oriented Computing: Proceedings of the 9th International Conference, ICSOC 2011, Paphos, Cyprus, Dec. 5-8, pp. 32–46.

Springer, Berlin, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25535-9_3
9. Pautasso, C.: BPMN for REST. In: Proceedings of the 3rd International Business Process Modeling Notation Workshop (BPMN 2011), pp. 74–87 (2011)
10. Mendling, J., Reijers, H.A., Recker, J.: Activity labeling in process modeling: empirical insights and recommendations. Inf. Syst. **35**(4), 467–482 (2010)
11. Leopold, H., Eid-Sabbagh, R., Mendling, J., Azevedo, L.G., Baião, F.A.: Detection of naming convention violations in process models for different languages. Decis. Support Syst. **56**, 310–325 (2013)
12. Miller, G.A.: WordNet: a lexical database for english. Commun. ACM **38**(11), 39–41 (1995)
13. Wu, Z., Palmer, M.: Verbs semantics and lexical selection. In: Proceedings of the 32nd Annual Meeting on Association for Computational Linguistics, pp. 133–138 (1994)
14. Resnik, P.: Using information content to evaluate semantic similarity in a taxonomy. In: Proceedings of the 14th International Joint Conference on Artificial Intelligence, pp. 448–453 (1995)
15. Lin, D.: An information-theoretic definition of similarity. ICML **98**, 296–304 (1998)
16. Kolb, P.: Disco: a multilingual database of distributionally similar words. In: Proceedings of KONVENS-2008, Berlin (2008)
17. Kolb, P.: Experiments on the difference between semantic similarity and relatedness. In: Proceedings of the 17th Nordic Conference on Computer Linguistics (2009)
18. Reiter, E., Dale, R.: Building applied natural language generation systems. Nat. Lang. Eng. **3**(1), 57–87 (1997)
19. Denger, C., Berry, D.M., Kamsties, E.: Higher quality requirements specifications through natural language patterns. In: IEEE International Conference on Software—Science, Technology and Engineering, pp. 80–90 (2003)
20. Leopold, H., Mendling, J., Polyvyanyy, A.: Generating natural language texts from business process models. In: Proceedings of the 24th International Conference on Advanced Information Systems Engineering, pp. 64–79 (2012)
21. Leopold, H., Mendling, J., Polyvyanyy, A.: Supporting process model validation through natural language generation. IEEE Trans. Softw. Eng. **40**(8), 818–840 (2014)
22. Knöpfel, A., Gröne, B., Tabeling, P.: Fundamental modeling concepts. Effective Communication of IT Systems, Wiley, England (2005)
23. Palma, F., Gonzalez-Huerta, J., Moha, N., Guéhéneuc, Y.G., Tremblay, G.: Are restful apis well-designed? Detection of their linguistic (anti)patterns. In: Service-Oriented Computing. Lecture Notes in Computer Science. Springer (2015)
24. Valverde, F., Pastor, O.: Dealing with rest services in model-driven web engineering methods. V Jornadas Científico-Técnicas en Servicios Web y SOA, JSWEB (2009)
25. Schreier, S.: Modeling restful applications. In: Proceedings of the Second International Workshop on Restful Design, pp. 15–21. ACM (2011)
26. Laitkorpi, M., Selonen, P.: Towards a model-driven process for designing restful web services. In: IEEE International Conference on Web Services, pp. 173–180. IEEE (2009)
27. Decker, G., Kopp, O., Leymann, F., Weske, M.: Bpel4chor: extending bpel for modeling choreographies. IEEE Int. Conf. Web Serv. **2007**, 296–303 (2007)
28. Jordan, D., Evdemon, J., Alves, A., Arkin, A., Askary, S., Barreto, C., Bloch, B., Curbera, F., Ford, M., Goland, Y., et al.: Web services business process execution language version 2.0. OASIS Stand. **11**, 1–10 (2007)
29. Alonso, G., Casati, F., Kuno, H., Machiraju, V.: Web Services. Springer, Berlin (2004)
30. Mendling, J., Hafner, M.: From WS-CDL choreography to BPEL process orchestration. J. Enterp. Inf. Manag. (JEIM) **21**, 506–515 (2008)
31. Kavantzas, N.: Web services choreography description language (ws-cdf) version 1.0. http://www.w3.org/TR/ws-cdl-10/ (2004)
32. Ouyang, C., Dumas, M., Van Der Aalst, W.M.P., Ter Hofstede, A.H.M., Mendling, J.: From business process models to process-oriented software systems. ACM Trans. Softw. Eng. Methodol. **19**(1), 2–37 (2009)
33. Ziemann, J., Mendling, J.: EPC-based modelling of BPEL processes: a pragmatic transformation approach. In: International Conference on Modern Information Technology in the Innovation Processes of the Industrial Enterprises, Genova, Italy (2005)
34. Mendling, J., Lassen, K.B., Zdun, U.: On the transformation of control flow between block-oriented and graph-oriented process modelling languages. IJBPIM **3**(2), 96–108 (2008)
35. Weber, I., Xu, X., Riveret, R., Governatori, G., Ponomarev, A., Mendling, J.: Untrusted business process monitoring and execution using blockchain. In: La Rosa, M., Loos, P., Pastor, O. (eds.) Business Process Management: Proceedings of the 14th International Conference, BPM 2016, Rio de Janeiro, Brazil, Sept. 18–22, pp. 329–347. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-45348-4_19
36. Omohundro, S.: Cryptocurrencies, smart contracts, and artificial intelligence. AI Matters **1**(2), 19–21 (2014)
37. Navigli, R., Ponzetto, S.P.: Babelnet: the automatic construction, evaluation and application of a wide-coverage multilingual semantic network. Artif. Intell. **193**, 217–250 (2012)
38. Medina-Mora, R., Winograd, T., Flores, R., Flores, F.: The action workflow approach to workflow management technology. In: Proceedings of the 1992 ACM conference on Computer-supported cooperative work, pp. 281–288. ACM (1992)
39. Cohen, W.W., Carvalho, V.R., Mitchell, T.M.: Learning to classify email into "speech acts". EMNLP **4**, 309–316 (2004)
40. Pittke, F., Leopold, H., Mendling, J.: Automatic detection and resolution of lexical ambiguity in process models. IEEE Trans. Softw. Eng. **41**(6), 526–544 (2015)
41. Weidlich, M., Mendling, J., Weske, M.: Efficient consistency measurement based on behavioral profiles of process models. IEEE Trans. Softw. Eng. **37**(3), 410–429 (2011)
42. Leopold, H., Niepert, M., Weidlich, M., Mendling, J., Dijkman, R., Stuckenschmidt, H.: Probabilistic optimization of semantic process model matching. Bus. Process Manag. **7481**, 319–334 (2012)

Adriatik Nikaj has obtained his Master's degree in Computer Science at the University of Paderborn, Germany. Currently he is a research assistant in the business process technology research group at Hasso Plattner Institute at the Digital Engineering Faculty, University of Potsdam, Germany. His research topics include business model of networked enterprises, business process choreographies, RESTful interactions, and blockchain technologies. His research papers are published in the Springer LNCS series.

Mathias Weske is chair of the business process technology research group at Hasso Plattner Institute at the Digital Engineering Faculty, University of Potsdam, Germany. The research group aims at addressing real-world problems in business process management with formal approaches and engineering useful prototypes. His research focuses on the engineering of process oriented information systems, process choreographies, and event handling. Dr. Weske is author of the first textbook on business process management, and he held the first massive open online course on the topic in 2013. With Matthias Kunze, he published a textbook on behavioral models. He is on the Editorial Board of Springer's Distributed and Parallel Databases journal, Springer's Distributed Computing journal, and he is a founding member of the steering committee of the BPM conference series and, since September 2017, chair of the steering committee.

Jan Mendling is a full professor with the Institute for Information Business at Wirtschaftsuniversität Wien (WU Vienna), Austria. His research interests include various topics in the area of business process management and information systems. He has published more than 300 research papers and articles, among others in ACM Transactions on Software Engineering and Methodology, IEEE Transaction on Software Engineering, Information Systems, Data and Knowledge Engineering, and Decision Support Systems. He is member of the editorial board of seven international journals, member of the board of the Austrian Society for Process Management (http://prozesse.at), one of the founders of the Berlin BPM Community of Practice (http://www.bpmb.de), organizer of several academic events on process management, and member of the IEEE Task Force on Process Mining. His Ph.D. thesis has won the Heinz-Zemanek-Award of the Austrian Computer Society and the German Targion-Award for dissertations in the area of strategic information management.