Semi-automatic Derivation of RESTful Interactions from Choreography Diagrams

Adriatik Nikaj[1(✉)], Fabian Pittke[2], Mathias Weske[1], and Jan Mendling[2]

[1] Hasso Plattner Institute at the University of Potsdam, Potsdam, Germany
{adriatik.nikaj,mathias.weske}@hpi.de
[2] Institute for Information Business, WU Vienna, Vienna, Austria
{fabian.pittke,jan.mendling}@wu.ac.at

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 choreography diagram 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 no systematic engineering approach to design their collaboration. In this paper, we address this gap by defining a semi-automatic method for the derivation of RESTful interactions from choreography diagrams. The method is based on natural language analysis techniques to derive interactions from the textual information in choreography diagrams. The proposed method is evaluated in terms of effectiveness and considered to be useful by REST developers.

Keywords: Choreography diagram · RESTful interactions · Natural language analysis

1 Introduction

Traditionally, research in BPM has focused on internal processes of organizations. The trend towards more complex services extends BPM towards 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.

However, taking business interactions, modeled by choreography diagrams, 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 IT developers with

R. Schmidt et al. (Eds.): BPMDS/EMMSAD 2016, LNBIP 248, pp. 141–156, 2016.
DOI: 10.1007/978-3-319-39429-9_10

respect to the business process choreographies. While there is first research into bridging this gap [3], a methodical approach for deriving REST interactions from business process choreographies is missing.

In this paper, we address this research gap and present an approach that takes as input a standard BPMN choreography diagram and it generates as output a RESTful choreography. Our approach is based on natural language processing techniques, which use textual descriptions of the choreography task to map to the most suitable 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 REST choreography is used to derive code skeletons which facilitate the development of REST APIs.

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 BMPN 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 that is also a RESTful choreography (see next subsection). 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 assumption here is that all the participants are subscribed to the RMS

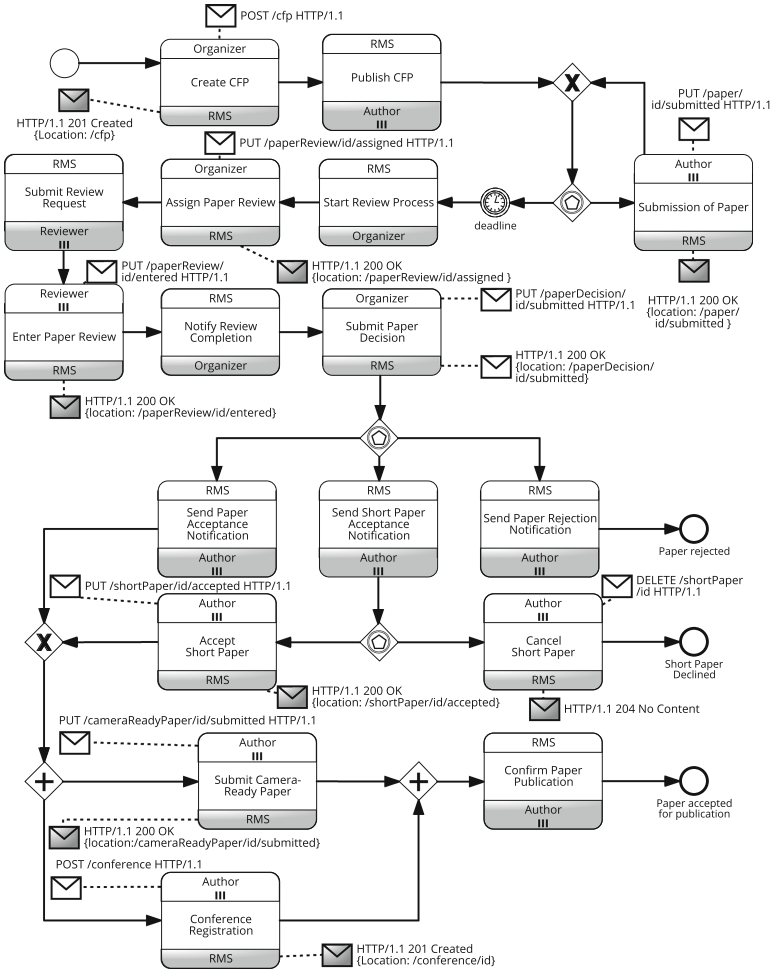

Fig. 1. RESTful choreography for paper submission and review management

and notified via email for any relevant information. 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 shows 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 grey. The same applies for the initiating and return messages (although the messages are not required to be graphically depicted). For example, the choreography task *Create CFP* has only the initiating message, while the choreography task *Submit Paper* has also a return message as a confirmation.

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.

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.

Business processes can be used to model the internal behaviour of the participants involved in RESTful conversation as proposed in [4]. However, when it comes to interactions between multiple participants, it is important to focus on a global perspective in order to reason about the state of common resources and the allowed interactions with these resources.

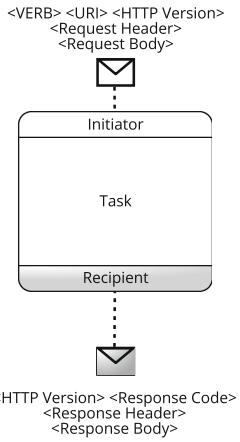

Fig. 2. The annotation of the choreograph task in RESTful choreography [3]

To this end, Nikaj et al. [3] 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 review* 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 1 depicts the RESTful choreography that can be manually derived by the same business process choreography (the choreography without the REST notation) as an input. However, the person responsible for the enhancement of the choreograph 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 then explains how the concepts are employed to identify the type of REST request that is expressed in a choreography task label. Section 3.3 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 our core artifact of our approach. 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 is a set of choreography tasks;
- E is a set of events;
- G is a set of gateways;
- S 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.

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 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 [5]. As an example, consider the label *Submit paper review* from Fig. 1. It contains the action *to submit* and the business object *paper review*. It is important to note that these components can be communicated in different grammatical variations. For instance, the labels *Paper submission* or *Conference registration* communicate the action in a different grammatical structure by using nouns, which express the actions *to submit* or *to register*, respectively.

In order to be independent of grammatical labeling structures, we rely on the label annotation approach of Leopold et al. [6] 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 review* from Fig. 1. According to the prior conceptualization, the action is given by α(Submit paper review) = *to submit* and the business object is given by β(Submit paper review) = *paper review*. 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 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 [7]. 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.

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 *Accept 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 accept* 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. [8–10]) to quantify this relatedness. We utilize the distributional similarity of the DISCO word similarity tool [11], denoted with sim_{DISCO}, because it outperforms existing similarity measures [12]. 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 2nd 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 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:

Table 1. Synonym word sets of the REST verbs

REST 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, [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, ...] }

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

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

3.3 REST Request Generation

The task of generating REST requests involves the generation of a 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 [13]. In contrast to these systems, we do not require a fully flexible approach, since the final links follow regular structures [3]. Therefore, we use a template-based approach [14–16] 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. Afterwards, 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 change is expressed by the past participle of the action of a choreography task label.

Table 2. Link templates for REST requests based on [3]

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>s	DELETE /shortPaper/id HTTP/1.1

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 form practice.

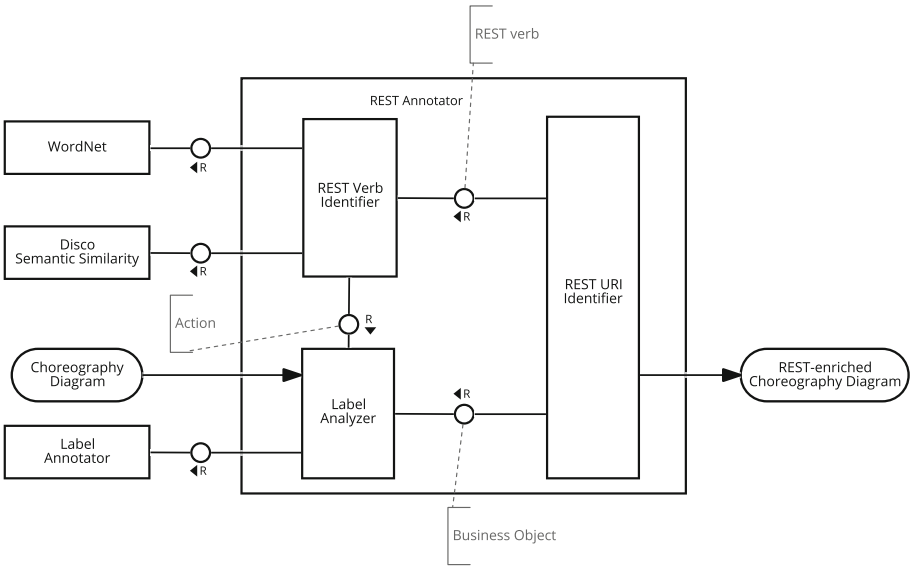

Fig. 3. REST Annotator software architecture

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. 3 as an FMC diagram [17]. 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. [6], WordNet [7], and the distributional similarity component of the DISCO tool [11]. 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 analysing 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 Extractor

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 set of choreography diagrams enriched with REST annotation.

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 have not been used for REST purposes, it is necessary to *clean the data*. In particular, we apply the following criteria:

1. *English-only Diagrams*. We include only diagrams with English text labels. This criteria is necessary because most of the natural language analysis components only support English.
2. *Non-trivial Diagrams*. We select those diagrams that describe a meaningful interaction between actors. In particular, we exclude diagrams with only one or two choreography tasks since they do not give sufficient context to judge their relevance for REST.
3. *Syntactically correct Diagrams*. Diagrams which have syntax errors with respect to the BPMN 2.0 choreography diagram specification are excluded.

After the cleaning, we ended up with a set of 172 choreography diagrams that satisfied all the criteria.

With regard to the *evaluation procedure*, we chose three human evaluators, who have extensive knowledge regarding REST APIs. The evaluators had to perform a three-step evaluation for each choreography task: (a) the REST relevance of a choreography task, (b) the correctness of the REST verb, and (c) the suitability of the generated REST URI. Their judgment on (a) is based on the contextual information they can obtain from the choreography diagram containing the task under assessment, e.g., the involved participants, the exchanged messages, the description of events, the entirety of all choreography tasks. This evaluation step is necessary because our test data contains task labels that do not describe an interaction at all. In case (a) holds true, the evaluator further has to rate if the identified REST verb is correct (b) and if the generated URI is suitable (c).

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. From these labels, 698 labels (57.54 %) actually describe a RESTful interaction and have been considered for the human assessment. In the following discussion, we only focus on those labels that are relevant for the REST context 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 REST-relevant Labels	698 (57.54 %)
No. of REST-irrelevant Labels	515
Total No. of Correctly Identified REST Verbs	523 (74.93 %)
.. with the Synonym Identification Strategy	265
.. with the Similarity Identification Strategy	258
Total No. of Incorrectly Identified REST Verbs	175 (25.07 %)
.. requiring a human decision among alternatives	55
.. requiring full human correction	120
Total No. of Correct Links	424 (60.74 %)
.. POST Links	56
.. PUT Links	182
.. GET Links	170
.. DELETE Links	16
Total No. of Incorrect Links	274 (39.26 %)
.. POST Links	26
.. PUT Links	76
.. GET Links	143
.. DELETE Links	29

The *verb identification strategies* have identified the correct REST verb in 523 labels which amounts to almost 75 % of all REST-relevant 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 to derive the correct REST verb in 265 labels, while the similarity strategy derives the correct REST verb for 258 choreography labels. The results emphasize the need for the similarity identification strategy of the REST verb. In total, 175 choreography labels (25.07 %) have been annotated with the wrong REST verb. We identify two classes of errors that lead to the wrong annotation. The first class subsumes choreography labels for which the similarity strategy revealed two or more equal similarity scores. This has been the case for 55 choreography labels. Here, our approach does not make a decision for one particular alternative, but it presents all alternatives to the user for selection. The second class covers such cases in which our approach identified the wrong REST verb. The evaluation has revealed 120 choreography labels for which our approach did not find the correct REST verb. These cases have also to be corrected by the user.

The approach to *generate RESTful links* has created 424 correct and 274 incorrect links. The results for the POST and PUT links are satisfactory because the amount of correct cases is clearly outnumbering the amount of incorrect one. For example, 56 POST links have been generated correctly, while 26 links are

incorrect. In case of the GET links, the ratio of correct and incorrect links is balanced. However, the evaluation showed that the DELETE links have issues in terms of correctness. 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* or *invoice sent*, or by not recognizing the business object correctly, e.g. *payment report* or *customer invoice*. Nevertheless, we conclude that the link generation works satisfactory and produces a large number of correct REST links.

We also exemplify the results by applying our approach to the example brought in Sect. 2. Figure 1 shows the derived RESTful choreography after the human correction is applied to the generated choreography from our tool. The REST Annotator generates correct REST URIs for 7 out of 9 REST-relevant 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 *accept short paper* and generates the URI *PUT /shortPaper/id/accepted*. 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 consequence, we require human interaction to choose among the possibilities. 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. The third observation covers such links that are incorrect and that need to be manually corrected by the user. As an example, consider the choreography task *conference registration*, for which the our approach creates a *GET* link. However, we would expect a *POST* or a *PUT* request. Incorrect links of this type may have several error sources. On the one hand, the label annotation component might have misclassified the choreography task and erroneously changed action and business object. On the other hand, the 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 URI.

At last, Fig. 4 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

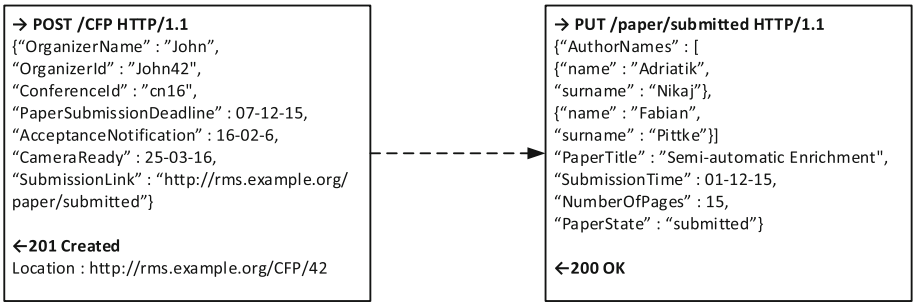

Fig. 4. A concrete skeleton instance of RMS implementation

the concrete instances of the *Create CFP* and *Paper Submission* choreography tasks from Fig. 1. 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. 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 [18]. The automation of deriving skeletons from a RESTful choreography is left as a future work.

5 Related Work

We identify two 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 [19] or Schreier [20], 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. [21] 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 model with its underlying orchestration system. With this regard, Decker et al. [22] propose an extension of BPEL web service composition standard [23] 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 [24]. Opposite to that, we take a top-down approach for deriving RESTful interactions. Another approach establishes the relation between BPMN and REST [4]. 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 perspective, 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.

6 Conclusions and Future Work

The paper defines a semi-automatic approach for deriving RESTful interactions from BPMN Choreography Diagram. 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. 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 three REST experts. They agreed that the verb identification is correct in 74.93 % of cases, while the URI is correct in 60.74 % of cases. This work contributes an additional step towards the research gap between business process choreographies and their implementation.

Our approach also has limitations in terms of the imprecise nature of natural language and the capabilities of the employed language processing tools. These issues mainly lead to wrongly identified REST verbs and incorrectly generated REST URIs that have to be corrected by users. In future work, we plan to make use of word sense disambiguation technology and of behavioral aspects of the choreography diagram. The former explicitly considers the semantics of words in a given context and improves the quality of the synonyms and the accuracy of semantic similarity. The latter relates to the sequential order of choreography tasks and might be helpful to resolve conflicts with several alternatives. 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.

References

1. OMG: Business Process Model and Notation (BPMN), Version 2.0. http://www.omg.org/spec/BPMN/2.0/
2. Fielding, R.T.: Architectural Styles and the Design of Network-based Software Architectures. PhD thesis (2000)

3. Nikaj, A., Mandal, S., Pautasso, C., Weske, M.: From choreography diagrams to RESTful interactions. In: Service Oriented Applications, WESOA 2015, co-located with ICSOC 2015. Springer (2015)
4. Pautasso, C.: BPMN for REST. In: Dijkman, R., Hofstetter, J., Koehler, J. (eds.) BPMN 2011. LNBIP, vol. 95, pp. 74–87. Springer, Heidelberg (2011)
5. Mendling, J., Reijers, H.A., Recker, J.: Activity labeling in process modeling: empirical insights and recommendations. Inf. Syst. **35**(4), 467–482 (2010)
6. 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)
7. Miller, G.A.: WordNet: a lexical database for English. Commun. ACM **38**(11), 39–41 (1995)
8. 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)
9. 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)
10. Lin, D.: An information-theoretic definition of similarity. In: ICML, vol. 98, pp. 296–304 (1998)
11. Kolb, P.: Disco: a multilingual database of distributionally similar words. In: Proceedings of KONVENS 2008, Berlin (2008)
12. Kolb, P.: Experiments on the difference between semantic similarity and relatedness. In: Proceedings of the 17th Nordic Conference on Computational Linguistics (2009)
13. Reiter, E., Dale, R.: Building applied natural language generation systems. Nat. Lang. Eng. **3**(1), 57–87 (1997)
14. Denger, C., Berry, D.M., Kamsties, E.: Higher quality requirements specifications through natural language patterns. In: 2003 IEEE International Conference on Software - Science, Technology and Engineering, pp. 80–90 (2003)
15. Leopold, H., Mendling, J., Polyvyanyy, A.: Generating natural language texts from business process models. In: Ralyté, J., Franch, X., Brinkkemper, S., Wrycza, S. (eds.) CAiSE 2012. LNCS, vol. 7328, pp. 64–79. Springer, Heidelberg (2012)
16. Leopold, H., Mendling, J., Polyvyanyy, A.: Supporting process model validation through natural language generation. IEEE Trans. Softw. Eng. **40**(8), 818–840 (2014)
17. Knöpfel, A., Gröne, B., Tabeling, P.: Fundamental modeling concepts. Effective Communication of IT Systems, England (2005)
18. 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). doi:10.1007/978-3-662-48616-0_11
19. Valverde, F., Pastor, O.: Dealing with rest services in model-driven web engineering methods. In: V Jornadas Científico-Técnicas en Servicios Web y SOA, JSWEB (2009)
20. Schreier, S.: Modeling restful applications. In: Proceedings of the Second International Workshop on RESTful Design, pp. 15–21. ACM (2011)
21. 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)

22. Decker, G., Kopp, O., Leymann, F., Weske, M.: Bpel4chor: extending BPEL for modeling choreographies. In: IEEE International Conference on Web Services 2007, pp. 296–303 (2007)
23. 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 Standard **11**, 1–10 (2007)
24. Alonso, G., Casati, F., Kuno, H., Machiraju, V.: Web Services. Data-Centric Systems and Applications. Springer, Heidelberg (2004)