Modeling RESTful Conversations with Extended BPMN Choreography Diagrams

Cesare Pautasso[1], Ana Ivanchikj[1(✉)], and Silvia Schreier[2]

[1] Faculty of Informatics, University of Lugano (USI), Lugano, Switzerland
c.pautasso@ieee.org, ana.ivanchikj@usi.ch
[2] innoQ Deutschland GmbH, Monheim, Germany
silvia.schreier@innoq.com

Abstract. RESTful Web APIs often make use of multiple basic HTTP interactions to guide clients towards their goal. For example, clients may get redirected towards related resources by means of hypermedia controls such as links. Existing modeling approaches for describing RESTful APIs expose low-level HTTP details that help developers construct individual requests and parse the corresponding responses. However, very little attention has been given to high-level modeling of RESTful conversations, which abstracts the structure of multiple HTTP interactions. To address such issue in this paper we introduce an extension of the notation used in BPMN choreography diagrams. Its purpose is to represent concisely all possible interaction sequences in a given RESTful conversation.

Keywords: RESTful web services · Conversations · BPMN choreography · Modeling notation extension

1 Introduction

In traditional messaging systems, conversations involve a set of related messages exchanged by two or more parties [1,2]. Web services borrowed the notion of conversation [3] to indicate richer forms of interactions going beyond simple message exchange patterns [4]. As more and more Web services [5] adopt the constraints of the REpresentational State Transfer (REST) architectural style [6], conversations remain an important concept when reasoning about how clients make use of RESTful Web APIs over multiple HTTP request/response cycles [7].

In this paper we introduce an extended version of the choreography diagrams of the Business Process Model and Notation (BPMN) 2.0 standard [8, Chap.5]. Our goal is to provide a concise and yet expressive visualization of all possible interactions that may occur in a given RESTful conversation, in order to facilitate the communication among RESTful APIs' architects and developers. The extension emphasizes details found when using the HTTP protocol, such as hypermedia controls [9], headers and status codes. They are all relevant for defining the salient properties of the request and response messages composing a

D. Weyns et al. (Eds.): ECSA 2015, LNCS 9278, pp. 87–94, 2015.
DOI: 10.1007/978-3-319-23727-5_7

RESTful conversation. To illustrate the expressiveness of the proposed notation we model an example of a frequently reoccurring conversation.

The BPMN for REST [10] extension we have proposed earlier in 2011 targeted the modeling of RESTful Web service invocations from business process models and the invocation of resources published from within business processes. In this paper we target a different viewpoint focusing on the interactions between clients and resources, while abstracting away the processes that represent the internal logic of the two (or more) parties involved in the conversation.

The rest of the paper is structured as follows. In Section 2 we define the main properties of RESTful conversations. We survey related work in Section 3. We introduce the extension for BPMN choreography diagrams and use it to model a well known conversation in Section 4 and conclude in Section 5.

2 RESTful Conversations

REST is a hybrid architectural style, which combines the layered, client-server, virtual machine and replicated repository styles with additional constraints (i.e., the uniform interface, statelessness of interactions, caching and code-on-demand) [6]. As a consequence, interactions within a RESTful architecture are always initiated by clients. They send request messages addressed to the resources hosted on servers which are globally identified by Uniform Resource Identifiers (URIs). Requests are always followed by response messages, whose representation may change depending on the current state of the corresponding resource. Relationships between resources can be expressed and resources can refer clients to related resources. This way, URIs are dynamically discovered by clients. The mechanism whereby hyperlinks (or resource references) are embedded into resource representations or sent along in the corresponding meta data [11] is one of the core tenets of REST, known as Hypermedia.

RESTful conversations thus can be seen as a specific kind of message-based conversation defined by the following characteristics: 1. Interactions are client-initiated; 2. Requests are addressed to URIs; 3. A request message is always followed by a response, however there may be different possible responses to the same request message; 4. Hypermedia: responses embed related URIs, which may be used to address subsequent requests; 5. Statelessness: every request is self-contained and therefore independent of the previous ones; 6. Uniform Interface: there is a fixed set of request methods a resource can support. Furthermore, it is possible to distinguish safe or idempotent requests from unsafe ones.

These characteristics make it possible to share the responsibility for the conversation's direction between clients and servers. Servers guide the client towards the next possible steps in the conversation by choosing to embed zero, one or more related URIs as hyperlinks in a response. Clients may choose which hyperlink to follow, if any (they may decide to stop sending requests at any time). This way, clients decide how to continue the conversation by selecting the next request from the options provided by the server in previous responses. In general, clients can accumulate URIs discovered during the entire conversation or may

remember them from previous conversations. Zuzak et al. call this the *Link Storage* in their finite-state machine model for RESTful clients [12]. Additionally, responses may be tagged as cacheable and thus clients will not need to contact the server again when re-issuing the same request multiple times. The discussion so far assumes that servers are available and always reply to client's requests[1]. In case of failures, either due to loss of messages or the complete unavailability of servers, an exception to the response-request rule must be made.

3 Related Work

The necessity of conceptual modeling of interactions has resulted in different modeling language proposals such as Let's Dance [13] or iBPMN [14], and has led to the introduction of the Choreography Diagram in version 2.0 of the BPMN standard [8, Chap.5]. Since the main targeted domain of these languages is modeling interactions involving traditional Web services, their capability of depicting effectively and efficiently RESTful interactions is limited, which has motivated our work on extending the BPMN choreography. RESTful Conversations have been introduced in [7], where they are used as an abstraction mechanism to simplify the modeling of individual RESTful APIs making use of them. In this paper we model the conversations themselves, which in some cases, may span across multiple APIs. Whereas in [7] UML sequence diagrams are used to visually represent the selected sample of conversations, in this paper we use extended BPMN choreography diagrams, following the fast-growing adoption of the BPMN standard which became an ISO standard in 2013 (ISO/IEC 19510).

4 Extension for RESTful BPMN Choreographies

The graphical representation of all the possible interactions, that may occur as part of a RESTful conversation, facilitates its comprehension. While UML sequence diagrams can be a good starting point when dealing with simple conversations [7], they are limited in concisely presenting conversations that can follow alternative paths. Therefore we propose using BPMN choreographies to visualize RESTful conversations. They focus on the exchange of messages with the purpose of coordinating the interactions between participants [15, pg. 315], while at the same time showing the order in which the interactions may occur.

As Lindland et al. [16] claim in their framework for understanding the quality in conceptual modeling, a very important aspect of a modeling language is its domain appropriateness. Cortes-Cornax et al. [17] emphasize the same when evaluating the quality of BPMN choreographies. They state that "the language must be powerful enough to express anything in the domain but no more". Therefore, to render the BPMN choreography diagrams more concise when targeting the modeling of RESTful conversations, we propose minor changes to their notation.

[1] Servers may indicate their unavailability by sending responses carrying the 503 Service Unavailable status code.

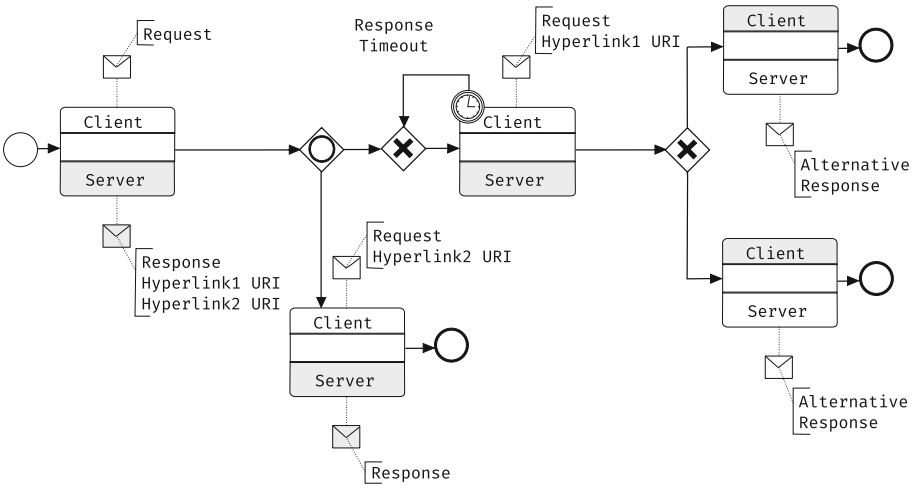

Fig. 1. RESTful conversation modeled with standard BPMN choreography

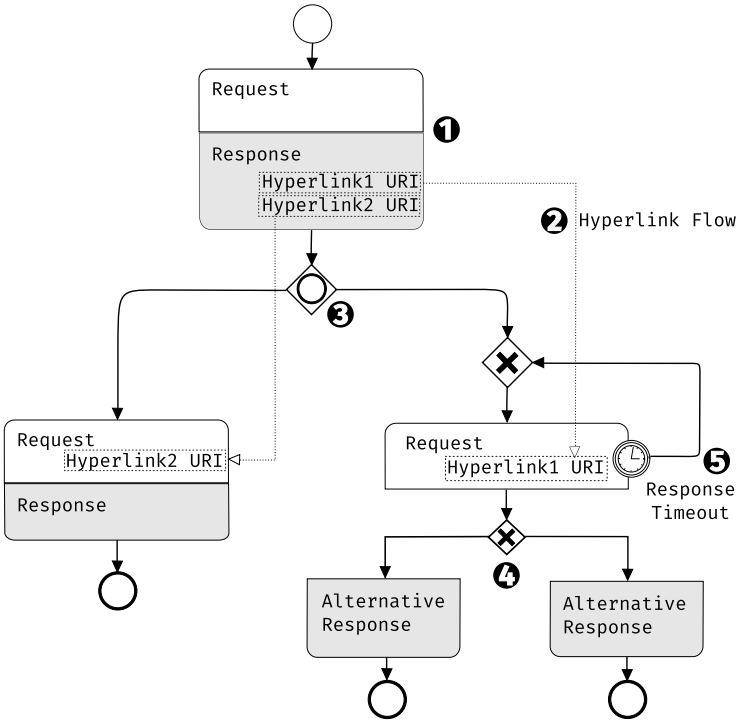

Fig. 2. RESTful BPMN choreography: proposed notation extension

As it happens often in high-level conceptual modeling [18, pg. 93], various assumptions and simplifications need to be introduced in order to avoid overwhelming the reader with too many visual elements. This usually results in

the exclusion of certain details from the models. We introduce the following list of assumptions and simplifications for the RESTful BPMN choreographies: 1. While a hyperlink that has been discovered by the client can be used at any time in the future, to avoid decreased readability due to line-crossing we only take into consideration the hyperlink from the last received response; 2. While clients may decide to stop sending requests at any time, we model a path as finished (by using an end event), only if an initially intended goal has been achieved; 3. While servers may send responses that include many different HTTP status codes, we only include the status codes which are relevant for the specific conversation. For example, 5xx status codes can occur at any time. The client will need to decide how to react to such errors depending on the domain and error details; 4. While clients may choose to resend idempotent requests an arbitrary number of times, we only model situations where the client retries sending non-idempotent request (POST, PATCH) after a *response timeout* event occurs.

Figures 1 and 2 show the same generic conversation in order to illustrate the proposed extension of the notation and its conciseness. In contrast to business processes where it is important to highlight which participant is responsible for initiating the interaction, in a RESTful conversation the initiator is always the client, and there is no one-way informative interaction. The content of the messages is of a particular interest, because it defines the action to be taken by the server and the future direction of the conversation. To comply with these differences, we replace the BPMN activity comprised of an optional incoming/outgoing message with a text annotation to depict the message content and a three band choreography task containing the names of the participants, with a two band request/response element with embedded message content (Fig. 2, no. 1). Moreover, since in RESTful conversations the focus is not on the activities but on their request/response content, we consider a vertical flow direction more intuitive to follow, with a starting event leading directly to client's request and the server's response leading directly to the following request or an end event.

The remaining extensions that we propose capture distinct tenets of RESTful APIs. The *hyperlink flow* indicates how URIs are discovered from hyperlinks embedded in the preceding response to clarify how clients discover and navigate among related resources (Fig. 2, no. 2). In RESTful conversations it is important to distinguish between: 1. Path divergence due to client's decisions, e.g., to navigate to a given resource or to end the process, in which case any type of gateway (exclusive, inclusive, parallel or complex) can be used (Fig. 2, no. 3); and 2. Path divergence due to different possible responses from the server to a given client's request, in which case only exclusive gateway can be used, since the server always sends exactly one response (Fig. 2, no. 4). In the latter case, the exclusive gateway is introduced between a given request and the alternative response messages. This is the only situation in which a request and its response are not aggregated in the same element. Response timeouts may occur when the server takes too long to respond and thus the client decides to resend the request. To model them we use an interrupting boundary timer event attached to

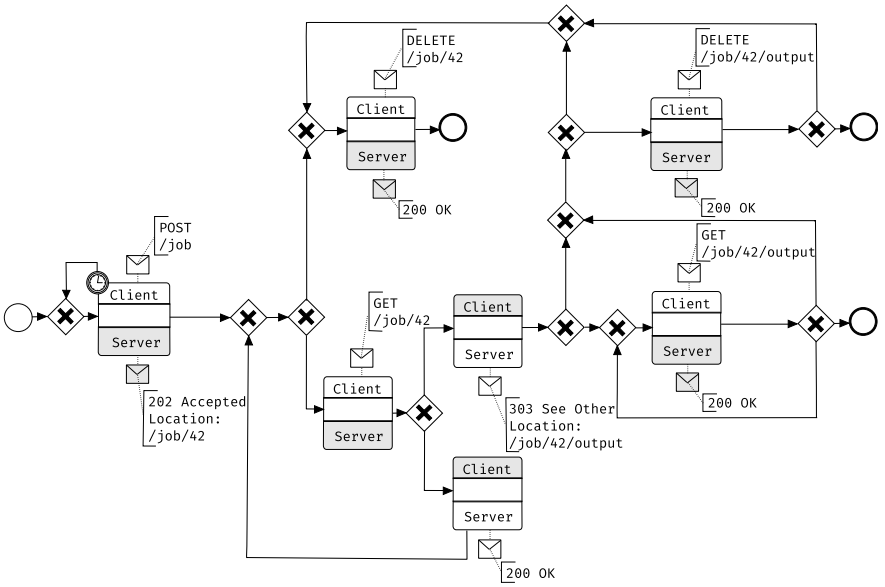

Fig. 3. Long running request modeled with standard BPMN choreography

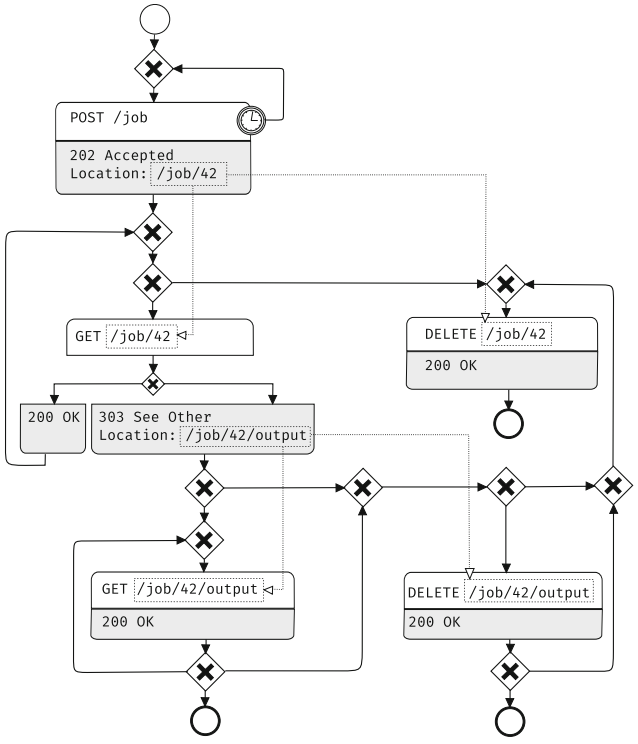

Fig. 4. Long running request modeled with the proposed extension

the request element. Such an event breaks the normal request-response sequence by introducing a request-timeout-request-response sequence (Fig. 2, no. 5).

In addition to the generic conversation shown in Fig. 1 and 2, in Fig. 4 we have applied our notation to a conversation that can be found in many RESTful APIs, e.g., Amazon Glacier's API for long term storage of infrequently used data[2]. Retrieving such data can take several hours (usually 3 to 5 hours [3]). Therefore to avoid having the client wait for such a long time, the operation is turned into a job resource, which is created using the original request. Assuming that creating the job twice has no side effects and the client does not receive a response to the job creation request within a given time frame, it can decide to send the POST request again. Once the job has been created the client may poll the job resource to GET its current progress and will eventually be redirected to another resource representing the output, once the long running operation has completed. Since the output has its own URI, it becomes possible to GET it multiple times, as long as it has not been deleted. Additionally, the long running job can be cancelled at any time with a DELETE request, thus implicitly stopping the operation on the server or deleting its output if it had already completed in the meanwhile. Fig. 3 and 4 show the conversation covering the whole lifecycle of a long running operation using the standard BPMN and our proposed extended BPMN notation, respectively. They illustrate how concisely this conversation can be visualized with our extension by emphasizing the important REST tenets.

5 Conclusion

Conversations are relevant in the context of RESTful Web APIs because multiple basic HTTP interactions are combined by clients navigating through the API's resources guided by the hyperlinks provided by the server. Thus, the design of RESTful APIs always consists of conversations and not only, for example, of the URI patterns and supported media types of its resources. Giving a visual representation of RESTful conversations is an important first step towards understanding and improving how RESTful APIs are designed.

The contribution of this paper is the graphical representation of RESTful conversations by proposing minimal extension to the standard BPMN choreography diagrams. The goal is to render the conversations more precise by focusing on the specific facets of RESTful APIs (e.g., hyperlink flow, request-response sequencing). We have illustrated the expressiveness of the proposed notation by modeling a typical conversation found in many RESTful APIs.

In the future we plan to design and conduct a survey among both designers of RESTful APIs and developers of client applications consuming them, to validate that the proposed notation enhances the understandability of RESTful conversations. Furthermore based on our experience and existing literature we plan to model a collection of frequently used RESTful conversation patterns and

[2] http://docs.aws.amazon.com/amazonglacier/latest/dev/job-operations.html
[3] http://aws.amazon.com/glacier/

to explore how to compose together individual reusable patterns to simplify the modeling of larger conversations.

Acknowledgments. The work is partially supported by the Hasler Foundation (Switzerland) with the Liquid Software Architecture (LiSA) project.

References

1. Hohpe, G.: Let's have a conversation. IEEE Internet Computing **11**(3), 78–81 (2007)
2. Barros, A., Dumas, M., ter Hofstede, A.H.M.: Service interaction patterns. In: van der Aalst, W.M.P., Benatallah, B., Casati, F., Curbera, F. (eds.) BPM 2005. LNCS, vol. 3649, pp. 302–318. Springer, Heidelberg (2005)
3. Benatallah, B., Casati, F., et al.: Web service conversation modeling: A cornerstone for e-business automation. IEEE Internet Computing **8**(1), 46–54 (2004)
4. Völter, M., Kircher, M., Zdun, U.: Remoting patterns: foundations of enterprise, internet and realtime distributed object middleware. Wiley, Chichester (2013)
5. Richardson, L., Amundsen, M., Ruby, S.: RESTful Web APIs. O'Reilly, Sebastopol (2013)
6. Fielding, R.T.: Architectural Styles and the Design of Network-based Software Architectures. PhD thesis, University of California, Irvine (2000)
7. Haupt, F., Leymann, F., Pautasso, C.: A conversation based approach for modeling REST APIs. In: 12th WICSA, Montreal, Canada, pp. 1–9. ACM, May 2015
8. Weske, M.: Business Process Management: Concepts, Languages, and Architectures, 2nd edn. Springer, Heidelberg (2012)
9. Amundsen, M.: Building Hypermedia APIs with HTML5 and Node. O'Reilly, Sebastopol (2011)
10. Pautasso, C.: BPMN for REST. In: Dijkman, R., Hofstetter, J., Koehler, J. (eds.) BPMN 2011. LNBIP, vol. 95, pp. 74–87. Springer, Heidelberg (2011)
11. Nottingham, M.: Web linking. Internet RFC 5988, October 2010
12. Zuzak, I., Budiselic, I., Delac, G.: A finite-state machine approach for modeling and analyzing RESTful systems. J. Web Eng. **10**(4), 353–390 (2011)
13. Zaha, J.M., Barros, A., Dumas, M., ter Hofstede, A.: Let's dance: a language for service behavior modeling. In: Meersman, R., Tari, Z. (eds.) OTM 2006. LNCS, vol. 4275, pp. 145–162. Springer, Heidelberg (2006)
14. Decker, G., Barros, A.: Interaction modeling using BPMN. In: ter Hofstede, A.H.M., Benatallah, B., Paik, H.-Y. (eds.) BPM Workshops 2007. LNCS, vol. 4928, pp. 208–219. Springer, Heidelberg (2008)
15. Jordan, D., Evdemon, J.: Business process model and notation (BPMN) version 2.0. OMG (2011). http://www.omg.org/spec/BPMN/2.0/
16. Lindland, O., Sindre, G., Solvberg, A.: Understanding quality in conceptual modeling. IEEE Software **11**(2), 42–49 (1994)
17. Cortes-Cornax, M., Dupuy-Chessa, S., Rieu, D., Dumas, M.: Evaluating choreographies in BPMN 2.0 using an extended quality framework. In: Dijkman, R., Hofstetter, J., Koehler, J. (eds.) BPMN 2011. LNBIP, vol. 95, pp. 103–117. Springer, Heidelberg (2011)
18. Robinson, S., Brooks, R., Kotiadis, K., Van Der Zee, D.J.: Conceptual modeling for discrete-event simulation. CRC Press Inc., Boca Raton (2010)