BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Excel@FIT 2016
http://excel.fit.vutbr.cz

# Business Process Representation as RESTful Resources

Alena Chernikava*

**Abstract**
Almost every company in the world deals with business processes on a daily basis. And business can derive significant benefits from taking a formal approach. This means, that the business process is formally described (for example using Business Process Modeling Notation 2.0) and implemented in some Business Process Engine (BPE) . BPEs have a limitation: they do not provide a possibility to change some ”part” of the process on the fly. So, all business processes are designed in a linear way and are not flexible. In order to change the business process, you need to hire a specialist, who will analyze and implement your requirement, stop the factory, reload the new business process and start the factory again. Usually factories are so tied to the current BPE that it is too expensive to adopt new technologies or to change the currently used BPE for another one. The aim of this paper is to design a general flexible RESTful API which provides more flexibility to the processes and is not so tied to the one concrete BPE, but makes possible to benefit from using many of them.

**Keywords:** REST, workflow, business process, RAML

**Supplementary Material:** RESTful API description using RAML

*, *Faculty of Information Technology, Brno University of Technology*

## 1. Introduction

In this paper, we provide a general description of RESTful API for business process engines (BPE; also known as workflow management systems), software frameworks for execution and maintenance of process workflows. Contrary to state-of-the-art approaches, the approach presented in this paper allows to change business processes on the fly (to enable dynamic business processes executions), to execute non-linear and context-aware processes (the processes that permit different executions based on their context, such as currently available resources), and to easily switch and integrate several underlying BPE platforms to efficiently execute and maintain complex business processes decoupled on several components in distributed environments.

Each company in the world wants to overcome its competitors and one of the ways for an efficiency improvement is to optimize company’s business processes. A *business process* (BP) is a collection of related, structured activities that produce a specific service or product for customers. The business can derive significant benefits from taking a formal approach. This approach consists in a formal description of the BP using the well-known common tools (e.g. BPMN 2.0 notation [1]) and its implementation using some BPE (e.g. Bonitasoft).

Every BPE is a comprehensive application incapsulating a big variety of different tools and even though there are some limitations. For example there is no easy way to change some ”part” of the BP on the fly. That is why all BPs should be completely described and all possible required variations should be included. Even though we cannot foresee all possible variations, and some flexibility would be very useful. Usually factories are so tied to the BPE, that it is too expensive to adopt new technologies or to change the current BPE for another one. There is a need to have one more abstraction level that should hide some low level details and provide an opportunity to change BPE at minimal cost.

What can inspire people more than a success of

some idea or its implementation? For example we can say, that the The World Wide Web fulfills the REST principles [2]. Everybody knowns, that WWW is a highly scalable, flexible, distributed computing platform. And its great success drives efforts of applying REST principles in Business Process Modeling (BMP). One important research paper about REST-style workflows is [3] where every activity is represented by a resource (activity-centric modeling paradigm). However, the main problem of direct applying of REST principles to BPM is an unmanageable explosion of number of resources in a process. Another approach but with information-centric modeling paradigm was taken in [4]. Authors managed to eliminate the problem of large number of resources, but introduced another not obvious one. There is an inconsistency between a visual representation of the BP (it is still activity-centric BPMN) and the way of controlling the business process (information-centric). We are going to take completely different approach. As a resource we will represent a business process with its variations and allow users to use BPMN.

A delay in the business process can be a cause of a big financial loss, so it is critically important to be able to adjust the running BP current conditions. Some attempts in [5],[6] where done, but they both require big changes and do not allow to "reuse" already modeled BPs. In order to fulfill the business needs we provide a description of a general RESTful API that allows to adopt the running BP on the fly to the current situation without an expensive total redesign of the BP and that provides one more abstraction layer.

Let us start from a BP representation. Under the BP we understand a BP designed in a special way by means of one existing BPE. Therefore, for us this is a black box (we are not aware about all internal details), but we are aware that there is some missing information which we call a (*hole*) (Figure 1).

This means that the process itself is almost determined but some components are variable (and this is already known at the early stage of the modeling). It is important to admit, that even if a component is variable, it should satisfy some known boundary conditions.

In order to create an additional abstraction layer we need to introduce a template of the process, a pattern of the process (a filled template) and an instance of the process. A pattern represents a ready to use model of the process therefore it saves operator's time, because he can use preselected variable components. This structure allows to generate BPs from their abstract definitions and provides enough flexibility for distributing the process across different BPEs.

The main contribution of our approach is the ability to implement dynamic business processes executions and, in general, to improve scalability and reliability of the business process engines maintaining the executions or their parts in distributed environments.

## 2. BPM and BPE

It is important to understand what BPEs are responsible for and what are the benefits from using them. A formalization of a BP is a way to enforce a control of BP implementation in a company and also allows to apply the same approach everywhere across the company. This leads to a cost reduction of the BP implementation and provides an easy way for the BP optimization.

A formal model breaks each BP into tasks, providing a way to see how well they are designed and performed. Also in the formal model should be described an order of task invocation, conditions under which tasks must be invoked, task synchronization and a dataflow.

The formalization of the BP starts with its model. A *Business Process Modeling* (BPM) is the activity of representing processes of an enterprise in a graph-like structure that is easy to visualize [7]. Such visual models are used to share a great deal of information (the internal details about the business process) with a wide variety of audiences (business analysts, project managers, usual employees, factory engineers and etc). Without a standard modeling technique, it is impossible to use BPM. So, the Object Management Group (OMG) has developed a standard called *Business Process Model and Notation* (BPMN).

The main goal of BPMN is to provide an intuitive and simple way to describe, model, implement and use various BPs. It is important, that BPMN provides a visualization of business oriented notions through the graphical elements (Events, Activities, Gateways, Data, Messages, Sequence and Message Flows etc).

The key to success of the company is to manage its BPs in a right way. Since late 1970s there have been developed a big variety of *Workflow Management Systems* (WFMSs) or *Business Process Engines* (BPEs). The systems, that provide an ability to model, report on, execute and dynamically control BPs involving humans and automated systems.

## 3. REST Architectural Style

We are already familiar with the problem and we can start with a solution that is going to be designed according the REST architectural style. In this chapter we introduce REST and shortly describe its basic idea and some of its principals.

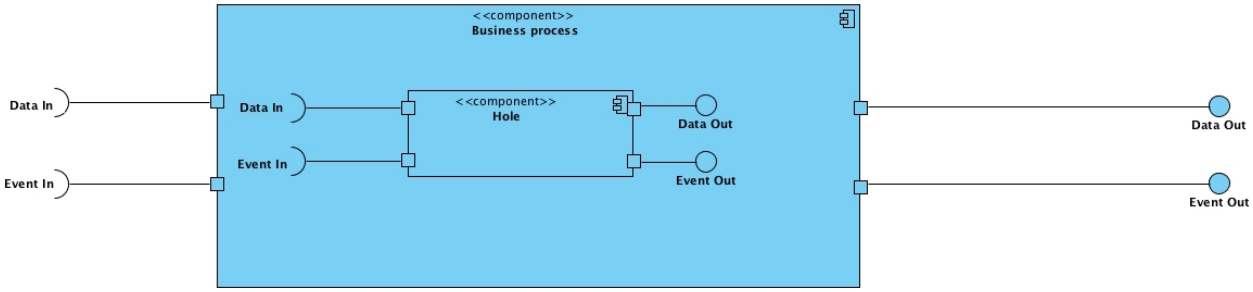


**Figure 1.** Visualization of the black box and hole idea.

REST states for *Representational State Transfer* and was designed in 2000 by Roy Thomas Fielding for distributed hypermedia systems [8]. The REST architectural style is not a standard, is not a protocol and is not an architecture. It is just a set of recommendations or principals. The REST principles are general and it is possible to apply them in different areas, but they found their main appliance at a web service and API design.

The basic idea is to define a list of entities and their representations. Each operation is a manipulation with an entity instance with help of a defined list of operations that could be performed under this particular entity.

The fundamental principle is a usage of a **client-server** communication pattern. Under the **stateless** principle we understand that a request sent to a server must contain all necessary information for its processing. The **uniform interface** principle states that between a client and a server a communication contract should be established. This contract consists of four conditions:

**Resource Identification** Every resource in the system should have a unique identifier (URI).

**Manipulation with Resource** A representation of the resource is sufficient for its update, removal and any other manipulation.

**Self-descriptive Message** Every message besides raw data must contain some meta-data.

**HATEOAS** (Hypermedia as the Engine of Application State) People tend to underestimate the importance of this point. First of all, a state of the resource can be changed by the client only through the list of possible actions received from the server. This means, that the model of the entity is complete, and client is able to select one from the provided possibilities and is not supposed to ”create” some new actions. The next requirement is that, a server but not a client specifies an URI namespace and the server should provide to the client information how to build an URI. Also it says, that a client communicates with a server through a dynamically generated hypermedia. This implies, that there is almost no need for a client to ”manually” create URIs. A server is supposed to provide different URIs to the client as part of resource representation.

## 4. RESTful API Design

Let us describe the basic ideas used for a design of our RESTful API. Web services designed according REST principles are called *RESTful APIs* and usually they are implemented on a top of the HTTP protocol [9]. The client sends a HTTP request (URI, headers, body) and the server responds with HTTP response (headers, body, HTTP return code). Four operations provided by HTTP protocol are used to manipulate with resources through their URIs:

**GET** provides read-only access to a resource
**PUT** modifies a resource
**DELETE** deletes a resource
**POST** creates a resource

To fulfill the HATEOAS principle every representation should contain some links to possible actions under the resource and some links to near resources.

As for us a BP is a black box with some missing information (*hole*) we need a deeper understanding of it. So it is possible to treat a hole as place for a sub-process, where only little information is known:

- What are the input data?
- What is the input event?
- What are the output data?
- What is the output event?

As it was already stated our solution would have three entities:

**Template** of the process is an entity, which represents a model of the process. Almost all internal details are hidden and some variable components are represented as holes. The process that does not have any holes can be also described with this template; just list of holes would be empty.

**Pattern** of the process (a filled template) is intended to represent a predefined set of process variations where variable components got concrete values.

**Instance** of the process is actually generated from some pattern. It represents an ongoing process. An instance can have states described on the (Figure 2)

As RESTful API is a tool for manipulating with resources we need to describe them. We choose JSON (Java Script Object Notation) as a tool for resource representation because it is a structured, lightweight, human- and machine-readable format.

To design some API is easy, but make its users really love it is harder. To make this happen we need to have easy-to-read documentation, API should be consistent, scalable and easy-to-test. All these targets we can achieve with *RESTful API Modeling Language* (RAML). It is a language for API description. As RAML is human friendly format it is easy to use. As RAML is a machine-readable we can easily generate a code prototype for a server's side and some mockups a client's side and furthermore, it is easy to generate a full documentation.

In this article there is no space to put entire description, so we will discuss some main points and the whole raml description for RESTful API you can find in the additional material.

As RESTful API is based on the resources and every resource must have a unique identifier (URI) we will start from the defining a general URI namespace (relative to the base URI):

- /templates
- /templates/{templateId}
- /patterns
- /patterns/{patternId}
- /patterns/{patternId}/holes
- /patterns/{patternId}/holes/{holeName}
- /instances
- /instances/{instanceId}
- /instances/{instanceId}/holes
- /instances/{instanceId}/holes/{holeName}

The structure is clear. Other details (method verbs, request and response body, returned HTTP codes and etc) you can find on github.

Here we will describe some basic ideas that lay behind that.

**Idea 1**. We do not want our users (clients) to generate URIs every time they want to communicate with the server. Being a server we want to provide a list of possible actions, that clients as supposed to use. Therefore every resource has a property "links" where the most common actions are described. Let ask ourselves a question "What actions a user is supposed to take under some instance?". As the user can do just a few actions (abort, run or pause) we will provide a direct instructions (URI + method) how to do that.

**Idea 2**. Usually assumptions are wrong. So in case we are not aware of some "specific" action, we will provide a user a possibility to manually build an URI by propagating "id" property into every resource representation.

**Idea 3**. To make our users happy we need to provide and easy way for a resource manipulation. This means, that a representation they obtain through the "GET" method they can reuse in "PUT"/"POST" methods. Generally it is one of the basic ideas of the REST, but during the design people underestimate this point, that leads to a need of redesigning and reimplementing the solution.

**Idea 4**. We cannot guarantee that description of the holes in the template and in the pattern will stay the same for ever. That is why every pattern and instance have its own copy of each hole description.

**Idea 5**. We can say, that a "hole" is not a part of "pattern" or "instance", but a separate resource. Also we should admit, that boundary conditions for the "hole" are described in a template and we cannot change them. The only thing that we can change is an assigned pattern. That is why we do not have an operation "update" the hole for patterns and instances, but instead of it we designed an action "assign pattern" in a list of possible actions under the resource.

**Idea 6**. Instance of the business process is quite complex. And we need to provide a simple way for sharing the information about its status. That is why, every instance provide a list of activities that are currently in progress (because some activities can be parallel) with information if this activity is "hole" or some "internal" activity of the process, a status of the activity and a list of resources that are required but are not currently available.

## 5. Conclusions

In this paper we discussed the current needs of business in an area of a business process management. Afterwards we introduced a possible solution to the problems in the form of RESTful API.

The designed API allows to change business processes on the fly. In the RESTful API we provided one more abstraction layer which helps to keep away BPE. Such approach opens an opportunity to change underlaying BPE without extremely high expenses.

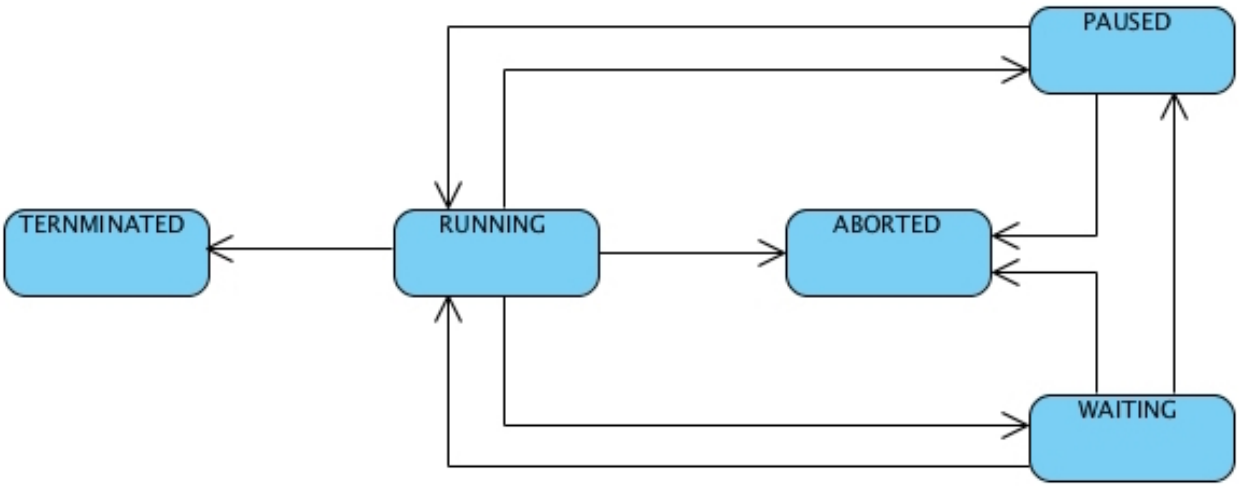


**Figure 2.** Possible state changes for instance

The main contribution of our approach is the ability to implement dynamic business processes executions and, in general, to improve scalability and reliability of the business process engines maintaining the executions or their parts in distributed environments.

In this paper we present only a description of RESTful API. The next step is to implement and test it with different BPEs and improve the current design if needed.

## Acknowledgements

I would like to thank my supervisor Marek Rychly for his help.

## References

[1] Documents associated with business process model and notation (bpmn) version 2.0. omg, 2011.

[2] Puting soap to rest. whitepaper, 2015.

[3] M. Muehlen, J. V. Nickerson, and Swenson K. D. Developing web services choreography standards – the case of rest vs. soap. *Decision Support Systems*, 40(1):9–29, July 2005.

[4] KUMARAN S., LIU R., DHOOLIA P., HEATH T., NANDI P., and PINEL F. A restful architecture for service-oriented business process execution. In *2008 IEEE 10th International Conference on e-Business Engineering*, pages 197–204, 2008. ISBN 978-0-7695-3395-7.

[5] In-Chul Song Joo Hyuk Jeon Sanghyun Yoo, Yohan Roh. Rule-based dynamic business process modification and adaptation. *International Conference on Information Networking*, pages 1–5, 01 2008.

[6] Yannis Chamodrakas-Drakoulis Martakos Nancy Alexopoulou, Mara Nikolaidou, editor. *Enabling On-the-fly Business Process Composition through an Event-based Approach*, 2008.

[7] D. GEORGAKOPOULOS, M. HORNICK, and A SHELH. An overview of workflow management: From process modeling to workflow automation infrastructure. *Distributed and parallel Databases*, 3(2):119–153, 1995. ISSN 1573-7578.

[8] Roy Thomas FIEelding. *Architectural Styles and the Design of Network-based Software Architecture*. PhD thesis, UNIVERSITY OF CALIFORNIA, IRVINE, 2000.

[9] ALLAMARAJU S. *RESTful Web Services Cookbook: Solutions for Improving Scalability and Simplicity*. O'Reilly Media, Inc, 2010. IBSN: 978-0-596-80168-7.