Abstract—We propose an extension of the Open API specification (OAS) for the semantic description of RESTful services. We suggest that API descriptions (e.g. on Clouds) be semantically annotated by associating OAS entities to entities in an ontology. The new approach, referred to Semantic OAS (SOAS) eliminates ambiguities and produces descriptions of RESTful services that are understandable by both, humans and machines. Moreover, SOAS descriptions are equivalent (i.e. can be translated) to an ontology. Therefore, SOAS services can be easily discovered (i.e. by other services), queried by Semantic Web query languages (e.g. SPARQL), checked by reasoners (e.g. Pellet) for consistency or inferring new (hidden) relationships and, finally, be used by service orchestration mechanisms.

Index Terms—Web services; cloud computing; ontology; Open API; REST;

I. INTRODUCTION

The focus of this work is on improving the description of API services, and consequently allow for the implementation of efficient and accurate service discovery mechanisms on the Cloud. API services need to be understandable by machines and be described in a way that eliminates ambiguities and provides descriptions which are uniquely defined and discoverable (e.g. by other services). Since the majority of Cloud services are offered by means of REST architecture [1], we adopt Open API Specification (OAS) as the main description language for Cloud services. OAS is a complete framework which is supported by a large set of tools. In order for a machine to understand the meaning of OAS, a service description need to be formally defined and its content be semantically enriched.

This work shows how OAS service descriptions can be semantically annotated by associating OAS entities to entities of an ontology (e.g. domain ontology). The new approach, henceforth referred to as Semantic OAS (SOAS) eliminates ambiguities in the original OAS descriptions and produces service descriptions that are understandable by both humans and machines. Moreover, this work proposes a mechanism for transforming SOAS descriptions to ontologies. This enables application of state-of-the-art query languages (e.g. SPARQL) for service discovery and of reasoning tools for detecting inconsistencies and inferring relationships between services. For a detailed description of the SOAS approach please refer to [2].

Open API specification and the proposed SOAS approach are discussed in Sec. II and in Sec. III respectively. The mechanism for translating SOAS service descriptions to ontologies is presented in Sec. IV followed by conclusions and issues for future work in Sec. V.

II. OPEN API SPECIFICATION (OAS)

The Open API Specification provides a basic format for describing RESTful services. The specification offers a syntactic service description that is enriched with text descriptions (through the extensive use of description property) so that users may easily discover and understand the functionalities of the service and interact with it. An OAS description is structured in terms of objects the most important of them being: The Info Object provides non-functional information about the described service, such as the name of the service, contact information with service provider, and terms of the service. The SecurityDefinitions Object contains the security schemes that the service uses for authentication. The specification offers support for basic authentication, an API key and OAuth2’s common flows. The Paths Object contains the relative paths for the service endpoints. Each Path item describes the available operations based on HTTP methods. The core of an OAS service document is the Operation Object. It contains all the needed information in order to construct HTTP requests to the service. In addition, it provides information regarding the HTTP responses that the service returns. The Responses Object contains all the expected responses of an Operation Object. It maps an expected response to a specific HTTP status code, describing the message content and HTTP Headers that an operation’s response may contain.

The Definitions Object contains the data types (called Schema Objects) that are used to describe the request and response messages. A Schema Object can be a primitive (string, integer) an array or a model. For the definition of Schema Objects the specification is based on JSON Schema.

References

1. https://github.com/swagger-api
2. https://www.openapis.org
and introduces additional properties (such as composition and polymorphism) and also the description of XML data types.

The Parameters Object contains the parameters that operations use and categorizes parameters into:
- **Path parameters** are used in cases where the parameter’s value is part of operation’s path.
- **Query parameters** are appended to the URI when sending a request.
- **Header parameters** define additional custom headers that may be sent in a request.
- **Body parameters** define the message content of a service request. A Body parameter refers to the data types defined in Definitions Object to describe the expected structure of the payload.
- **Form parameters** are a special case of Body parameters, that describe the content of a service request when the media types “application/x-www-form-urlencoded” and “multipart/form-data” are used as the content type of the request.

Listing 1 demonstrates how polymorphism support is provided using the discriminator property: A Person model uses the “gender” property as a discriminator in order to determine whether this is a model of a male or a female person.

```
Listing 1. OAS Model polymorphism example

definitions:
Person: # A Person model that will be extended
type: object
properties:
  firstName:
    type: string
  lastName:
    type: string
gender:
  type: string
required:
  - firstName
  - lastName
  - gender
discriminator: gender
Male: # A Male model extending the Person Model
description: A representation of a male person
allOf:
  - $ref: '#/definitions/Person'
    type: object
    properties:
      height:
        type: integer
        description: height in cms
      weight:
        type: integer
        description: weight in kgs
required:
  - height
  - weight
Female: # A Female model extending the Person Model
description: A representation of a female person
allOf:
  - $ref: '#/definitions/Person'
    type: object
    properties:
      eyesColor:
        type: string
required:
  - eyesColor
```

The x-refersTo extension property specifies the association between an OAS element and a concept in a semantic model. The property accepts a URI, that represents the concept in the semantic model. Listing 2 demonstrates how the property x-refersTo is used to semantically annotate a Person model and its properties. However, it is not always possible for a model to have a relation with an equivalent semantic concept, as a model may have a narrower meaning over the referenced semantic concept. For example, consider the Person model in Listing 2. If the model was defined to describe a specific group of people (e.g. teenagers), it would be inappropriate to associate the model with a generic concept such as the Person type of the Schema.org vocabulary, whereas a more specific subtype (if existed) should be considered more appropriate. In this case, the x-kindOf extension property is used denoting that the model is actually a subclass of the referred semantic concept. The property accepts a URI, representing the concept in the semantic model, and is only used for models in Definitions Object.

```
Listing 2. Semantic annotations of Schema Objects
definitions:
Person:
  type: object
  x-refersTo: http://schema.org/Person
  properties:
    firstName:
```
The \textit{x-mapsTo} extension property is used to define OAS elements that share the same semantics. In a OAS service document, there are many elements that share the same semantic information that a human may understand but a machine cannot. In the following, for ease of discussion, we refer to the Swagger Petstore service description. There is a semantic similarity among the \textit{petId} property of the Order model and the \textit{id} property of the Pet model. In addition, this semantic similarity also exists between the \textit{status} property of the Pet model and the \textit{status} query parameter. A human may easily infer these semantic similarities, either by the element names or by the description that may be provided. However, in order for a machine to act similarly to a human it is necessary to provide additional information, that specifies these relations. In SOAS, this is achieved with the \textit{x-mapsTo} property.

The \textit{x-collectionOn} extension property is used to indicate that a model in Definitions Object is actually a collection. A collection (or a list) is defined in OAS using the \textit{array} type. However, it is very common a collection’s definition to be enriched with additional properties and thus to be defined as an \textit{object} type. Using the \textit{x-collectionOn} property a model is considered to be a collection.

The \textit{x-onResource} extension property is used in Tag Objects in order to specify the resource that a tag refers. In OAS, tags are used to group operations either by resources or any other qualifier. If the tag is used to group operations by resources, a human may recognize that the referred resource is described by a Schema Object in Definitions Object. This information a machine is not able to infer and thus the \textit{x-onResource} property is introduced and associates the tag with a Schema Object that describes a specific resource. Schema Objects can be semantically annotated using the \textit{x-refersTo} property and thus refer to a semantic content with a specific meaning. Listing 3 demonstrates the usage of the \textit{x-onResource} property on the Swagger Petstore service description. The property is assigned on a \texttt{pet} tag that provides information regarding the operations that are available for the resource that is described by the Pet model in Definitions Object.

Finally, the \textit{x-operationType} extension property is used in order to semantically specify the type of an Operation object. A request (an operation for OAS) is characterized by the HTTP method that it uses. However, the semantics of the HTTP methods are too generic and in the context of a service they may have a more specific meaning. For example, in the Swagger Petstore service description a GET request on the path \texttt{/pet/findByStatus} (Listing 3) is a search operation of pets based on their status. A human may refer to the description of the operation in order to understand its intended purpose, but a machine needs additional information as the HTTP method itself cannot provide such specific information. This information is specified by the use of the \textit{x-operationType} extension property. The value of the property is a URL pointing to the concept that semantically describes the operation type. The \texttt{Action} type of the Schema.org vocabulary provides a detailed hierarchy of \texttt{Action} subtypes that can be used by the property.

\section*{IV. THE OPEN API ONTOLOGY}

SOAS extension properties allow an Open API document to become machine-understandable. Following this, an Open API document can be transformed to an ontology so that the representation of a service can benefit from Semantic Web tools (e.g. reasoners, query languages) enabling service discovery and service orchestration. Fig. 4 illustrates the structure of the proposed Open API ontology. The \texttt{Document} class represents the documentation and the entry point of the service. Similarly to the OAS structure, it provides general information (\texttt{Info} class) regarding the service, and also specifies the service paths and the entities it supports. A service path is described by the \texttt{Path} class containing the relative path for the service endpoint (using the \textit{pathName} property). The \texttt{Operation} class corresponds to the \texttt{Operation object} in the OAS structure.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{open_api_ontology.png}
\caption{Structure of the Open API ontology.}
\end{figure}
It contains information for expressing HTTP requests to the service as well as its responses.

The Security class defines the security schemes that the specification supports. The Operation class refers to the defined security scheme through the SecurityRequirement class, which in the case of the OAuth2 security scheme contains the scopes that are needed in order to successfully execute the operation.

Schema Objects can be associated with semantic concepts; it is feasible to express them in terms of classes, object and data properties and also capture all the information provided from the Open API document. The Entity class represents the classes that describe the models of an Open API service description and captures all the information contained in the models they describe. It specifies operations that are related to a class (supportedOperation) associated with the x-onResource extension property. In addition, it determines whether a class may contain additional properties (additionalProperties) of a specific type, described by a TemplateProperty and specifies the properties that a class contains (supportedProperty).

The Property class represents the properties that a class definition contains. A Property specifies the supported property (predicate), its expected value (valueType) and also possible restrictions (e.g., a maximum value for a numeric property). In addition, Property class indicates whether the supported property is required or not. The Array class is introduced in order to denote properties that contain a list of values (properties of type “array). Listing 4 illustrates how the “Person” model of Listing 2 is represented in the ontology.

Collections are represented in the ontology by the Collection class by specifying the members (member) of a collection. A “PetCollection” class is defined in the Open API ontology as a subclass of the Collection class. The “PetCollection” is also considered an Entity in order to describe its properties.

OAS parameters are represented as classes for every parameter type. In Fig. 1 Header class, contains all the definitions of header parameters that are used in HTTP requests and responses. Body class defines the body parameters describing the expected message content of a HTTP request. FormData class represents the form parameters that are defined by the specification describing the message content of a HTTP request when the “application/x-www-form-urlencoded” and “multipart/form-data” media types are used. Parameter class defines all parameters that are attached to operation’s URL. The class is further organized in PathParameter and QueryParameter classes that refer to the corresponding path and query parameters of the specification. Parameters of an Open API service description (excluding body parameters) are represented in the Open API ontology similarly to a model’s property definition.

V. CONCLUSION

We propose SOAS, an extension to Open API specification and we show that the yielding descriptions of RESTful services are equivalent to an ontology. SOAS descriptions are machine understandable so that they can be easily discovered by Semantic Web query languages (e.g., SPARQL), checked by reasoners (e.g., Pellet) for consistency and used by service orchestration mechanisms.

REFERENCES